Санкт-Петербургский государственный университет

Кафедра Технологии Программирования

Смирнов Максим Тимофеевич

Дипломная работа

Прогнозирование финансовых временных рядов нейронными сетями

Научный руководитель:
ст. преп. А. В. Уланов

Санкт-Петербург
2016

Содержание

Введение

В течение значительного времени люди, принимающие решения, испытывали желание, а зачастую также и потребность в ускорении такого принятия решений (этот термин будет пониматься буквально и очень широко, охватывая большую часть человеческой когнитивной деятельности).

Со временем для описания алгоритмов[1] такого принятия решений были разработаны математические области и дисциплины, которые часто тесно связаны друг с другом. Несколько позже, с появлением вычислительных машин, эти алгоритмы были перенесены на них, что привело к бурному развитию имеющихся научных областей и появлению новых.

В частности, во второй половине XX века стали развиваться области машинного обучения и искусственного интеллекта (ИИ). Достижения, полученные в этих областях, по мере прогресса в них были адаптированы в области человеческой деятельности, в которой решения должны приниматься как можно быстрее, и к тому же они имеют существенные последствия – в области финансов. Финансовый анализ уже довольно долго применяет методы ИИ, включая многочисленные разновидности искусственных нейронных сетей (ИНС, ANNs). Однако до сих пор, если судить по доступным научным публикациям, в этой области было обращено мало внимания на нейроэволюцию. В этой работе предпринимается попытка исправить такое положение.

Нейроэволюция – метод машинного обучения, использующий эволюционные алгоритмы для обучения искусственных ИНС).

В этой работе показывается, что нейроэволюция как метод ИИ способна эффективно решать определённые проблемы в области финансового анализа.

[1] а также эвристик

Постановка и актуальность задачи

В современном мире существует значительное количество разных видов финансовых сущностей; дабы не углубляться в области экономики и финансов, они не будут здесь рассматриваться подробно. Важным для нас является то, что обмен этим сущностями может и уже довольно давно (с 1971, хотя настоящий рост начался примерно с 2010) происходит в электронном виде – это называется электронной торговлей [2]. Современные информационные технологии и средства связи позволяют продавцам и покупателям встретиться на электронных торговых площадках [3]. Эти же технологии приводят к значительному росту количества событий, происходящих в некоторую единицу времени в области электронной торговли. Такие массивы информации, а также общая сложность этой области сделали явной необходимость создания машин, занимающихся торговлей. Современные автоматические торговые системы [4] представляют из себя программные комплексы, воплощающие достижения в областях финансовой математики и анализа, теории принятия решений и управления рисками и многих других. Их использование позволяет выполнять повторяющиеся операции на несколько порядков быстрее, чем это могли бы делать человеческие исполнители.

Технический анализ работает как самореализующееся пророчество [5]. Идея интересная, но сложная, и заслуживающая подробного рассмотрения в отдельной работе. В этой работе, однако, используется следующее её понимание: поскольку люди (и машины, управляемые представлениями таких людей) считают, что в рыночных сигналах есть система [6], они там и в самом деле появляются. Если наша система способна замечать такие геометрические образы [7], и делать это быстрее и лучше других систем, то можно ожидать от её использования некоторых выгод.

[2] e-trading
[3] electronic/ online trading platforms
[4] automated trading systems
[5] self-fulfilling prophecy
[6] patterns in the market signals
[7] geometrical patterns

ИНС активно применялись и продолжают применяться на финансовых рынках; одним из основных достоинств ИНС, делающих их такими популярными в качестве предсказателей рынка, является их природная нелинейность, позволяющая им выучивать нелинейные отображения и корреляции данных. ИНС также работают на данных [8], могут быть обучены в реальном времени; они высокоадаптивны, легко переучиваются в случае колебаний рынка и, наконец, хорошо справляются с данными, содержащими какое-то количество ошибок. Нейросети устойчивы.

В английском используется понятий robust, которое будет переводиться как “устойчивы”.

Человек-трейдер, особенно если он придерживается технического анализа, при взгляде на графики смотрит обычно не на чистые списки цен, а на встречающиеся фигуры.

Тут предпринята попытка подобрать адекватный перевод английского pattern [53]. В этой работе они же могут называться “особенностями”, “системой” или как-нибудь ещё, если то, что имеется в виду, понятно из контекста.

Для некоторых из часто встречающихся фигур существуют устоявшиеся человеческие названия: “голова и плечи”, “чашка с ручкой”, “треугольники”, и так далее [51]. Есть ли в них подлинный смысл – вопрос спорный, ответ может зависеть как раз от природы самореализации пророчеств. Прагматический же их смысл в том, что если мы сможем вырастить агента, способного реагировать на подобные особенности быстрее других трейдеров, мы сможем эксплуатировать подобное рыночное поведение.

Классические ИНС, используемые для предсказания цен и смежных задач, используют в первую очередь так называемый метод скользящего окна [9].

Данные поступают на вход ИНС в виде простого вектора. С точки зрения исследователей нейроэволюции это недостаток такой сети: она не видит графиков, а значит, не использует данные, доступные трейдеру-человеку.

[8] ANNs are data driven

[9] sliding window approach

Используемый метод предполагает желательность того, чтобы выращиваемый ИНС агент умел “смотреть” на графики. Как это сделать? Поставлять нейросети картинки с графиками (например, в виде .bmp изображений) бесполезно; сеть в любом случае воспримет эти данные как вектор (одномерную последовательность чисел), к тому же всё равно не отображающий геометрические образы.

Решением является кодирование субстрата. Этот метод активно применяется в компьютерном зрении [10] и естественным образом помогает реализовать различение геометрических свойств сенсорных сигналов и выявить геометрические регулярности в данных.

С такими непрямо кодированными нейросетями можно анализировать графики цен непосредственно, используя геометрически образы и тренды в данных. Для достижения поставленных целей будут использованы TWEANNs (topology and weight evolving artificial neural networks, ИНС, эволюционно изменяющие топологии и веса) для выращивания агентов, которые будут не только предсказывать следующие значения цен, но и напрямую обращаться к поставщику фин. услуг, собирать данные о ценах и совершать сами торги.

Вообще говоря, можно было бы попробовать использовать для обучения каждого агента свой экземпляр[11] электронной торговой платформы, такой, как MetaTrader. Но это вычислительно тяжело и нерационально, даже с учётом непосредственных условий эксперимента.

Вместо этого мы используем исторические данные о ценах и создадим симулятор Форекса [12] на Эрланге. Для каждого агента из популяции будет выделен приватный ландшафт – симуляция Форекса, а использование для этой задачи Эрланга поможет лучше распределить вычислительную нагрузку.

Достоверность симуляции повлияет на качество выращиваемых агентов; при достаточно точной симуляции настоящих параметров будет возможно вырастить популяцию агентов, пригодных для использования вместе с реальными электронными торговыми платформами.

---

[10] computer vision, CV

[11] instance

[12] Forex, FX, Foreign Exchange – обмен иностранных валют

Обзор использованной литературы

Несмотря на то, что область ИИ и машинного обучения в целом, нейронных сетей и эволюционных методов в частности существует и развивается давно и содержит значительное количество литературы, область нейроэволюции, а в особенности её применения в области финансового анализа очень молода и начала развиваться недавно [1, 3].

В этой работе используется в первую очередь метод, разработанный Шером [Ibid]; теоретические обоснования цитируются по работам их иностранных авторов и исследователей. Многочисленные авторы излагают основы своих областей и высказывают (часто смелые) предположения о результатах, которые смогут быть достигнуты при использовании предлагаемых ими методов. Более поздние авторы излагают свои результаты исследования некоторых областей и попутно создают новые. Армстронг (во вступлении к [1]) возлагает большие надежды на сети процессорных ядер и реализацию на их основе эффективных нейронных сетей, предположительно способных привести к универсальному ИИ[13]. Другие авторы предлагают иные подходы как ведущие к тому же результату.

В этой работе не строится полноценный ИИ; на данный момент методом, изобретённым Шером, решается одна из задач в области финансового анализа.

[13] Artificial General Intelligence, [49]

Глава 1.

Метод решения поставленной задачи

Машинное обучение на финансовых рынках можно использовать двумя способами: либо использовать агента для предсказания будущей цены финансового инструмента и поставления этих данных человеческому агенту для торговли на основании этого предсказания, либо использовать агента также и для автономного трейдинга, без человеческого вмешательства. В этой работе был выбран второй путь.

Классические ИНС многократно показали свою применимость в области финансового анализа в силу своей высокоустойчивой природы [1] и способностей к универсальной аппроксимации [4–12].

В литературе преимущественно описаны ИНС, использующие обратное распространение ошибки (back propagation) [13–17].

Поскольку обратное распространение ошибки является алгоритмом локальной оптимизации, следующие ему сети могут "застрять" в локальном оптимуме, и с ними это иногда случается.

Хуже того, исследователю обычно необходимо задать топологию ИНС заранее, но при этом предсказать, какая именно топология должна быть у нейросети, соответствующей данному набору данных и данному рынку, очень сложно или даже невозможно[2]. В итоге исследователям приходится воплощать топологии наугад и затем испытывать их, прежде чем они

[1] highly robust nature

[2] "Невозможность" здесь имеется в виду в неформальном смысле; строгие доказательства вычислительной невозможности мне неизвестны.

смогут выбрать некоторую определённую подходящую сеть.

TWEANNs способны менять не только свои веса, но и свою топологию. Они избегают проблем обратного распространения ошибки и способны осуществлять устойчивый глобальный поиск [3]. Именно поэтому они так хороши в применении к финансовому анализу. К сожалению, как новая технология они пока мало и не очень тщательно исследованы.

В ходе дальнейших исследований будет построен симулятор Форекса, а также агенты на основе сетей с эволюционирующей топологией (TWEANNs), субстрат-кодированные (substrate encoded neural networks, SENNs), реализующих подход PCI (Price Chart Input), и "обычные", использующих прямое кодирование и PLI (Price List Input), и будет анализирована и сравнена их производительность.

Ожидается, что SENNs покажут лучшие результаты.

[3] robust global search

Глава 2.

Элементы теории искусственных нейронных сетей

Ричард Докинз (R. Dawkins) называет людей “машинами для выживания” [1], которыми пользуются гены для того, чтобы взаимодействовать с окружающей средой, успешно конкурировать с другими “машинами” и в итоге распространять копии себя, как и положено репликаторам [37].
В ходе “гонки вооружений” биологическая эволюция создала нейронные сети – объединения клеток, позволяющих лучше управлять “машинами для выживания”.

[1] survival machines

Биологические нейроны.

Рассмотрим, как устроен современный биологический нейрон.

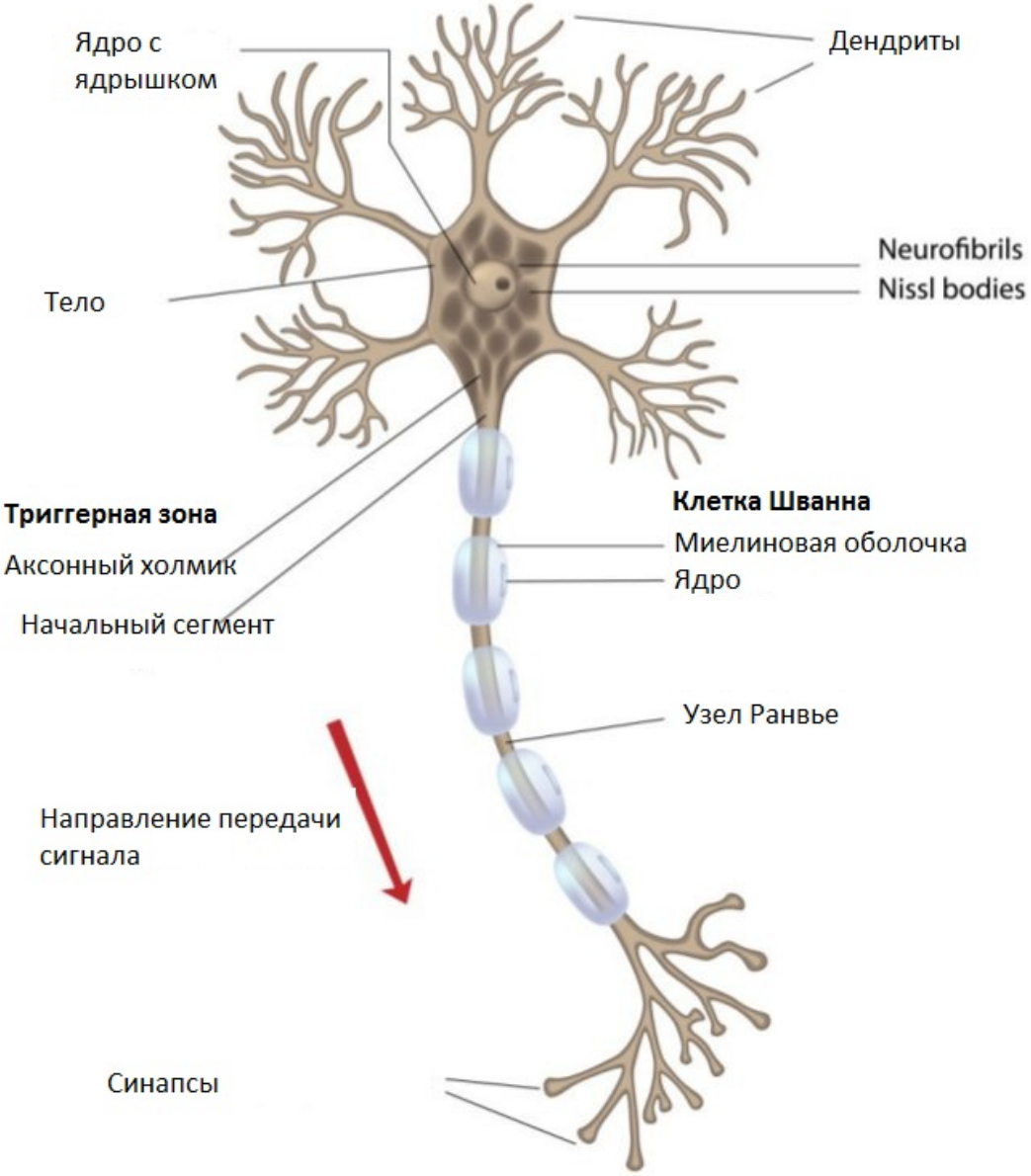

Рисунок 2.1 — Биологический нейрон. [1]

Нейрон – клетка, способная принимать сигналы и, в силу своих химических и геометрических свойств, производить вывод [2]. Типичный нейрон состоит из трёх частей: тела, аксона и дендритов (см. рис. 2.1). Дендритов у нейрона обычно много, аксон только один. Аксон обычно длинный; на некотором расстоянии от тела своего нейрона он начинает ветвиться и соединяться с другими нейронами. Дендриты и тело нейрона принима-

[2] output

ют синаптические сигналы, а аксон их передаёт. Синапс – контакт между дендритом или телом одного нейрона и аксоном другого.

Синаптические сигналы могут быть возбуждающими или тормозящими. Если общее возбуждение, полученное нейроном за короткое время, достаточно велико, он производит краткий импульс, называемый потенциалом действия, который быстро распространяется по аксону и активирует синапсы других нейронов [52].

Рассмотрим обработку сигналов нейронами подробнее. Нейрон посылает электрический импульс по аксону, окончания аксонов выделяют нейротрансмиттеры, воспринимаемые рецепторами на дендритах других аксонов, а это в свою очередь вызывает электрические импульсы в них.

В клеточных условиях очень сложно управлять величиной электрического потенциала и гораздо проще управлять частотой, с которой он подаётся[3].

Импульсы преимущественно воспринимаются аксонным холмиком. Вычисления (в биохимическом смысле) того, производить ли потенциал действия (также называемый "всплеск" [4]), зависят от формы нейрона и частоты, с которой импульсы поступают на аксонный холмик. Можно сказать, что в некотором смысле нейрон проводит пространственно-временное интегрирование входящих сигналов.

Со временем нейрон меняет параметры обработки сигналов: на дендритах становится больше или меньше рецепторов, они начинают работать иначе, а сам аксон ветвится, отсоединяясь от одних клеток и соединяясь с другими – это называют нейропластичностью.

Так происходит обучение и накопление опыта в биологической нейронной сети.

[3]Большая часть нейронов человеческого мозга работает на частоте примерно 20 Гц. На уровне ума это приводит к невероятному количеству "закешированных" мыслей. [18]

[4]spike

Искусственные нейроны.

Искусственные нейроны, вдохновлённые биологическими образцами, работают иначе. Отметим сразу, что в ИНС можно использовать частотное кодирование сигнала, так же, как в биологических нейронах [21, 22, 23]. В методе, использующемся в этой работе, однако, оно не используется. Рассмотрим теперь искусственный нейрон, считающийся более классическим[5].

Искусственные нейроны – абстракции биологических, созданные для моделирования их природных свойств: нелинейной обработки сигналов, пластичности и параллельности.

Нейрон принимает сигналы на своих входах, моделирующих дендриты, приписывает им весовые коэффициенты, пропускает взвешенную сумму через свою функцию активации, что соответствует пространственно-временному интегрированию на аксонном холмике, и затем распространяет выходящий сигнал по другим нейронам.

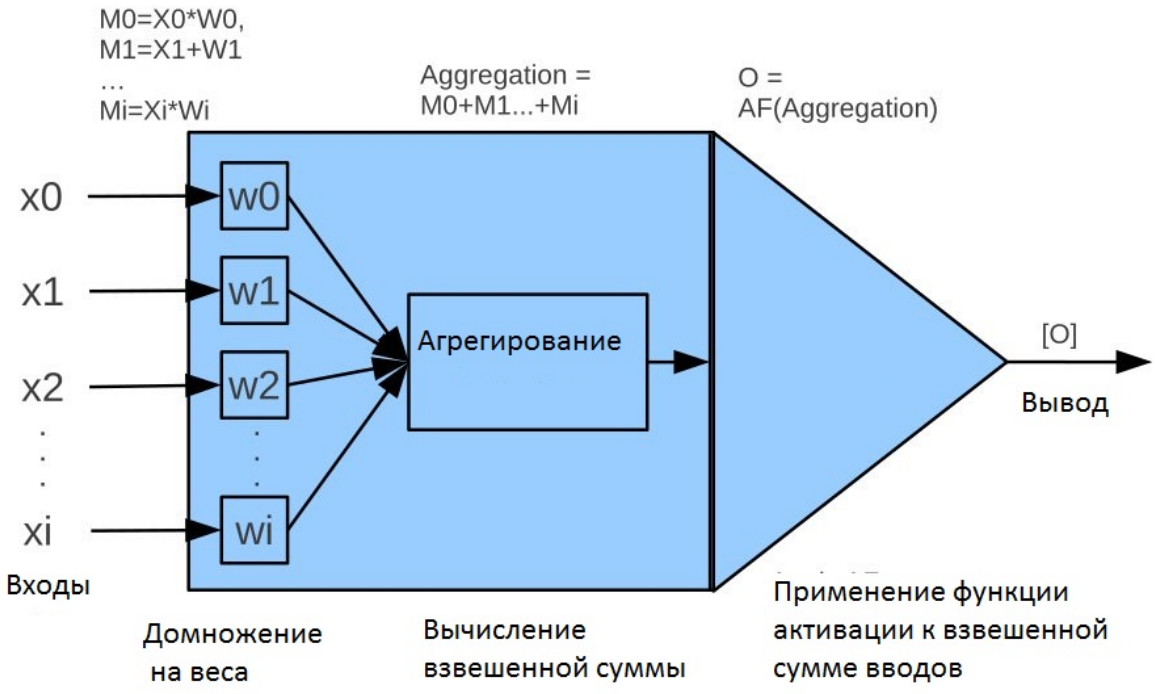

Рисунок 2.2 — Схема искусственного нейрона. [1]

Несмотря на все упрощения, функциональность нейрона сохраняется.

[5]Какое кодирование лучше: количественное или частотное? У чего больше гибкости?Сложно сказать. Из литературы известно, что и биологические, и искусственные НС полны по Тьюрингу [21]. Заметим только, что обработка частотно кодированных сигналов требует времени, количественно кодированный сигнал же нужен только один

Сети. Топология. Рекуррентность.

Нейрон сам по себе является простым обрабатывающим элементом; настоящая вычислительная мощь проявляется в системе, составленной из множества взаимосвязанных нейронов – искусственной нейронной сети.

ИНС обычно состоит из множества слоёв; их количество называют глубиной сети.

Различные организации слоёв и связей между нейронами в слоях называют топологиями сетей. Одно из основных делений ИНС по топологиям может быть на сети прямого распространения [6] и рекуррентные [7]. В рекуррентных сетях нейроны из некоторого слоя с номером x могут связываться с нейронами из слоёв с номерами меньше или равными x ; таким образом, нейрон в рекуррентной сети может принимать на вход собственный выход (см. рис. 2.3).

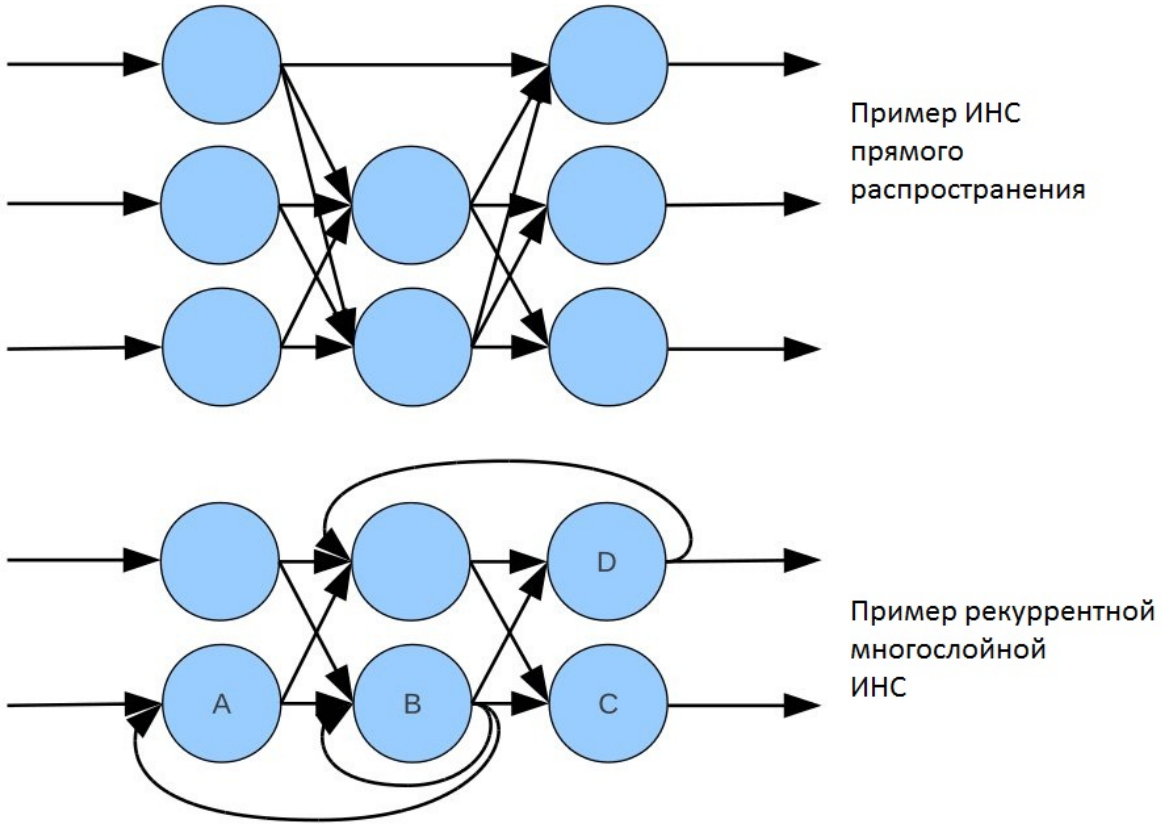

Рисунок 2.3 — Разные топологии ИНС. [1]

Чтобы накапливать опыт и улучшать свою производительность, ИНС должна меняться. Существует множество алгоритмов, по которым могут

[6] feedworward

[7] recurrent

происходить такие изменения, и их можно причислить к группам, называемым “обучение” [8] или “тренировка” [9]. Тренировка также известна как “обучение с учителем” [10]; важно присутствие некоторого внешнего алгоритма, оценивающего результаты и вносящего изменения. Метод обратного распространения ошибки относится к тренировкам.

В отличие от тренировок, обучение предполагает накопление опыта, и то, какой опыт получит нейросеть, во многом управляется ей самой. Обучение во многом опирается на нейропластичность.

В алгоритмах обучения с учителем предполагается выведение [11] некоторой целевой функции из множества тренировочных данных. Каждый тренировочный пример состоит из вектора входа и желаемого вектора выхода, также называемого обучающим сигналом [12]. В применении к ИНС обучение с учителем предполагает автоматизацию изменения весов в сети с использованием внешней системы, сравнивающей вывод сети с заранее известным верным выводом. Обучение с учителем возможно только в ситуациях, в которых есть множество тренировочных данных.

Одним из самых распространённых алгоритмов тренировки является алгоритм обратного распространения ошибки [13]. Рассмотрим его подробнее [22, 23].

Обратное распространение ошибки (ОРО) использует градиентный спуск для поиска минимальной функции ошибки (разницы) между выводом нейросети и желаемым выводом. Чаще всего этот алгоритм применяют к сетям прямого распространения (хотя вариант для рекуррентных сетей тоже существует).

Будем называть нейронной системой (или нейронным агентом) структуру, состоящую из 3 компонентов: нейронной сети, сенсоров, поставляющих сети сигналы о некотором окружающем её мире, и актуаторов, принимающих сигналы от сети и преобразующих их в воздействия на этот мир (см. рис. 2.4).

---

[8] learning
[9] training
[10] supervised learning
[11] inference
[12] supervisory signal
[13] error back propagation

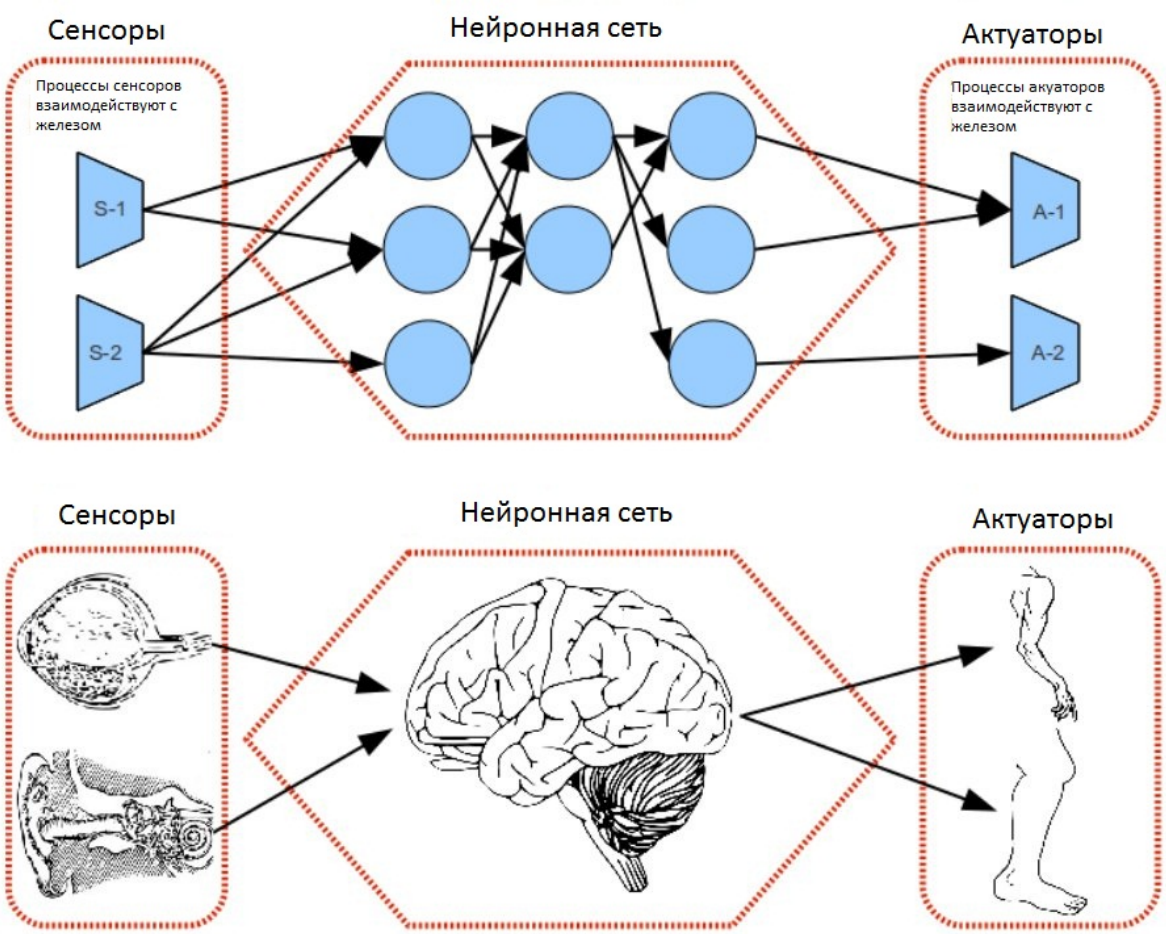

Рисунок 2.4 — Сенсоры и актуаторы. [1]

Глава 3.

Эволюционные методы

Методологическое замечание: когнитивные искажения заставляют людей антропоморфизировать биологическую эволюцию и считать, что она создала человеческий мозг намеренно. Но эта работа – по нейроэволюции и ИИ, а не эволюционной психологии людей. В любом случае, важно понимать, что эволюция, естественная или машинная, есть наблюдаемый оптимизационный процесс, и приписывать ему человеческие свойства эпистемически опасно для исследователя [18].

Биологическая эволюция в той версии, которой пользуются эволюционные биологи с середины 1960-х, очень сложна для понимания и как таковая в этой работе почти не рассматривается. Для наших целей важно понимание, которого эволюционные биологи достигают долго и аккуратно; эволюция – не человек, а оптимизационный процесс. Эффективность этого оптимизационного процесса в биологической области такова, что вдохновлённые люди попытались его формализовать и воспроизвести. Результатом исследований такого воспроизведения и стали эволюционные методы.

Введём некоторые определения:

Алгоритмы эволюционных вычислений – основанные на популяциях оптимизационные алгоритмы, вдохновлённые и выведенные из Дарвиновских[1] принципов биологической эволюции. Рассмотрим её подробнее, чтобы понять, как её суть может быть выражена и использованна программно.

Биологическая эволюция началась с появлением первого работающего репликатора и продолжилась в сторону увеличения сложности; в какой-то момент репликатор был скопирован неидеально, и это первичное отли-

[1] На самом деле неодарвинистских.

чие привело к началу конкуренции.

Для того, чтобы эволюция происходила, необходима популяция организмов/агентов и некоторый способ представления их геномов. Также необходим процесс отбора – некоторый метод, который позволит упорядочить организмы по мере их приспособленности (fitness).

В случае с эволюцией программ ни к чему усложнять агентов – рациональнее создавать их процессы отдельно от процесса, который будет управлять их оценкой и отбором.

Вводятся следующие определения: Генотип – совокупность генов, характеризующая организм. Фенотип – сам организм, проявление генотипа во внешней среде. Если представлять себе генотипы как находящиеся в пространстве поиска, а фенотипы – в пространстве решений, то станет ясно, что эволюционный алгоритм, отображающий генотипы в фенотипы – просто продвинутый поисковый алгоритм оптимизации.

Для того, чтобы эволюция могла происходить, необходимо, чтобы генотипы могли подвергаться изменениям; в случае программ за это будут отвечать операторы мутаций.

При решении конкретных задач организмы взаимодействуют со внешней средой. В случае биологических организмов ей является окружающий мир, для программ же средой является некоторая симуляция. Такие симуляции называются ландшафтами [2]. Агенты могут решать задачи самостоятельно, взаимодействуя с частными ландшафтами [3], либо они могут взаимодействовать друг с другом на общем ландшафте [4].

Когда задача известна, нужно решить, как именно представлять генотип, чтобы к нему было удобно применять операторы мутации. Генетическое программирование в том виде, который популяризировал Джон Коза, использует представление генотипов в виде деревьев [24, 25, 30, 31]. Преимущество деревьев в том, что их очень просто мутировать: можно добавить к дереву узел в случайном месте или заставить два дерева поменяться ветвями (в биологии такой обмен частями генотипа называют кроссинговер).

Некоторые эволюционные подходы приводят к “застреванию” в локальных

[2] scapes

[3] private scapes

[4] public scape

оптимумах, поэтому рассмотренный простой эволюционный цикл имеет множество разнообразных вариаций и дополнений, способных существенно ускорить поиск решений и итоговую приспособленность найденных агентов.

Рассмотрим основные из них: генетические алгоритмы, генетическое программирование, эволюционные стратегии и эволюционное программирование.

Генетическое программирование – один из наиболее известных подходов в эволюционных вычислениях. Компьютерные симуляции эволюции были известны с 1954, но популярны они стали с выходом книги Д.Холланда [30]. В своём подходе он подражает биологической эволюции и воплощает стандартные механизмы отбора, скрещивания, мутации и выживания наиболее приспособленных [5].

В основном алгоритм опирается на скрещивание; потомки создаются с помощью случайного выбора отрезков генотипа предков. Мутация используется в меньшей мере, для поддержания минимального разнообразия в популяции.

Проблема с классическими генетическими алгоритмами, однако, в том, что они используют неизменный размер генотипа. Но эта проблема была решена добавлением некоторых специфических операторов мутации, позволившим работать с генотипами, размер которых изменяется.

В генетических алгоритмах генотип обычно представляется в виде строки. Если же он представлен в виде дерева, то подмножество генетических алгоритмов, обарщающихся с ним, называется генетическим программированием.

Генетическое программирование также использует особый набор операций мутации, таких как обмен ветвями, мутации узлов и другие. Также, в отличие от классических генетических алгоритмов, размер деревьев в генетическом программировании может меняться.

Генетическое программирование было введено Крамером [31], но популяризировано Козой, который применил ГП к эволюции новых материалов (с помощью большого кластера, который Коза окрестил “Машиной Изобретений” [26]), антенн и других конструкций [27, 28].

[5] survival of the fittest

Эволюционные стратегии – направление, в котором рассматриваются изменения эволюционных параметров: вероятностей применения того или иного опреатора мутации, количества потомков, вероятностей выбора значения для узла дерева, представляющего генотип, и других. Таким образом можно создать не только популяцию генетически разных агентов, но и агентов, эволюционирующих с разной скоростью.

Эволюционные стратегии были введены Швефелем [29], а их исследование продолжил Рехенберг.

У этого направления есть свои преимущества; эволюции подвергается и сам процесс эволюции, подобно тому, как это происходит в природе. Эволюционное программирование – область поисковых алгоритмов, тесно связанная с эволюционными стратегиями, но разработанная раньше и независимо от них, и специализировавашаяся на эволюции таблиц перехода состояний для конечных машин состояний [6]. Эволюционное программирование было разработано Фогелем в 60-ые во время подъёма популярности искусственного интеллекта. Со временем о нём забыли в ИИ-сообществе, но затем его возродил Фогель-младший [32].

Кроме упомянутых, существуют также меметические алгоритмы, известные ещё как гибридные [7]. Это эволюционные алгоритмы, разделяющие эволюцию на две фазы: глобальный и локальный поиск. Глобальный поиск может создать большое разнообразие потомков в результате применения мощных операторов мутации; локальный же поиск позволяет этим потомком исследовать свои “окрестности” и найти там локальные оптимумы.

Было показано, что меметические алгоритмы исключительно эффективны, гораздо эффективнее стандартных однофазных [2, 33].

При использовании стандартных эволюционных алгоритмов можно иногда найти почти идеальный генотип, в котором нужно изменить всего несколько параметров. Но алгоритм скорее всего припишет ему низкую приспособленность, и он будет отброшен. Локальный поиск может помочь в тонкой настройке такого генотипа.

Меметические алгоритмы проявили себя особо при создании и выращивании ИНС – они являются одним из лучших решений в области нейроэволюции [3, 34, 35].

[6] finite state machines, конечные автоматы

[7] hybrid algorithms

Глава 4.

Нейроэволюция

Методы нейроэволюции многочисленны. Обычно различают методы, изменяющие нейронные веса при фиксированной топологии сети (общепринятая нейроэволюция / conventional neuroevolution) и методы, предполагающие изменения и топологии, и весов (TWEANNs, topology and weight evolving artificial neural networks, ИНС, эволюционно изменяющие топологии и веса).

Также методы делят в зависимости от того, как и когда они меняют сети: генетические алгоритмы меняют веса и топологии одновременно, меметические занимаются тонкой настройкой [1] весов внутри цикла, входящего в общий цикл изменения топологий.

Деление также проводится по таким критериям, как кодирование (encoding, оно может быть прямым или непрямым) и учёт нейропластичности. “Кодирование” означает отображение генотипа в фенотип [1]. При прямом кодировании каждая часть фенотипа описывается в генотипе явно; при непрямом генотипу необходимо пройти через некоторые циклы развития или использовать внешнюю информацию для построения фенотипа. В силу влияния среды на формирование фенотипа результат может оказаться специфичным для сложившихся условий. Непрямое кодирование часто используется с нейросетями, управляющими агентами в искусственной жизни (Alife, симуляция простых организмов, охотящихся друг на друга на плоскости (или в ином ландшафте)).

При субстратном кодировании (substrate encoding) генотип содержит списки, которые затем используются для построения нейронных сетей “сло-

[1] fine tuning

ями". Такое представление лежит в основе метода HyperNEAT [44]. Плотность субстрата может зависеть от среды или задаваться стохастически, что позволяет создавать очень плотные нейронные субстраты из тысяч нейронов, при этом описывая их кратко по сравнению с прямым описанием той же сети.

Исследования в области ИНС показали, что правило Хебба, самоорганизующиеся карты и другие методы, применяемые в ИНС, недостаточны; параметры в них приходится настраивать исследователю. Нейропластиности каждого отдельного нейрона также недостаточно; вся нейросеть должна быть настроена так, чтобы взаимодействие её нейронов приводило к осмысленному и желаемому результату. Таким образом, нам всё ещё необходим способ настройки различных параметров ИНС. Для сложных проблем и крупных сетей обратное распространение ошибки неэффективно, к тому же необходимо подбирать топологию таких сетей.

Решение состоит в использовании эволюционных алгоритмов для оптимизации нейронных сетей – это и есть нейроэволюция. Нейронные сети – графы. Всё, что необходимо исследователям, – разработать представление и кодирование генотипа и набор операторов мутации, достаточно мощный для перевода любого генотипа A в любой иной генотип B.

Существует множество путей для хранения графов. Есть возможность выбрать любой; строки, как в NEAT [35] , или иной подход, более дружественный к реляционным базам данных, используемый в DXNN и DXNN2 [1, 3]. Второй метод гораздо проще для человеческого понимания и применения операторов мутации, в нём вместо подражания биологическому геному используется гибкость, предоставляемая программно. Также, такое представление проще воплотить на Эрланге.

Сам процесс, отображающий генотип в фенотип, тоже может работать по-разному. В применяемом методе используется генотип, хранящийся в базе данных Mnesia, и фенотип, представленный процессами на Эрланге. Отображатель [2] в таком случае просто считывает пары из базы данных и создаёт процессы для каждого узла, используя считанные параметры[3].

[2] mapper

[3] И представление, и фенотип можно воплощать по-разному. Очень перспективными выглядят нейроморфные мемристоры (Knowm), но в их отсутствие мы используем программную реализацию.

Глава 5.

Нейроэволюционный метод решения

Обоснование выбора Эрланга как языка реализации. Его преимущества

Эрланг[1] как язык был создан для создания распределённых, основанных на процессах, ориентированных на парадигму передачи сообщений, устойчивых, в том числе и к ошибкам, параллельных систем.

Имеются в виду переводы robust (устойчивый к возмущениям), fault tolerant (устойчивый / "терпимый" к ошибкам) и concurrent (предполагающий много процессов, происходящих одновременно).

Для этого языка существует очень точное отображение на проблемную область разработки нейрокогнитивных систем. В этой главе это будет показано подробнее.

Выше было рассказано, как устроены ИНС. Но осталась проблема: то, как ИНС фнукционируют, их архитектура, способ обработки информации, количество требующейся им обработки, параллельность и распределённость нейросетей трудно отобразить в архитектурах стандартных языков программирования, таких как C/C++, C#, Java, Lisp и так далее.

[1]Официально заявляется, что Эрланг так назван в честь датского математика Агнера Крарупа Эрланга, исследователя трафика в телекоммуникационных системах и теоретика массового обслуживания (в сороковых годах в его честь была названа единица измерения трафика в телекоммуникационных системах). Но неофициально многие согласны, что Эрланг – просто сокращение от Ericsson Language.

Источником проблемы в данном случае является лингвистический детерминизм, вкратце постанавливающий, что сложно думать и создавать новые идеи на языке, который не был для этого разработан и не имеет соответствующих элементов. К примеру, на языке, не имеющем поддержки для математических концепций, будет очень сложно думать о математике, и скорее всего такая необходимость приведёт к революции в языке.

При программировании на C++ язык заставляет программиста думать о системах на C++ вместо того, чтобы думать о нейронных сетях, подлинной цели разработки. Архитектурно ИНС очень отличаются от используемых сейчас языков программирования. Для разработки устойчивых ИНС и возможности сосредоточиться на ИИ без необходимости беспокоиться о языке программирования нужно иметь инструменты, точно отображающие исследемую область; они позволять значительно упростить концептуальную сторону разработки и выражения идей. Именно такие инструменты предлагает Эрланг.

Если бы нам пришлось разрабатывать язык программирования конкретно для создания нейронных сетей, какие свойства ему бы потребовались? Конечно, хотелось бы точно отобразить в языке архитектуру нейронных сетей.

Соответственно, в языке должны быть структуры, схожие с сетями. Для этого потребовалось бы следующее:

1) Биологические нейронные сети состоят из независимых, параллельных, распределённых обрабатывающих единиц – нейронов. А тогда архитекутра нашего языка программирования потребует поддержки таких элементов, независимых процессов, которые могут выполняться параллельно и легко распределяться в современном железе.

2) Бионейроны общаются друг с другом посредством сигналом. Нашему ЯП (языку программирования) потребуется обеспечить общение процессов с помощью сообщений или сигналов.

   Отображения архитектуры, однако, недостаточно. Человеческие мозги устойчивы; мы нечасто попадаем в ситуации, в которых из-за ошибки в мозгу весь ум “падает”. Другими словами, наш мозг устойчив к ошибкам и самовосстанавливается. Нашему ЯП потребуется

поддержка такой же устойчивости:

3) Возможность лёгкого восстановления после ошибок.

4) Если какой-нибудь из элементов вычислительного интеллекта выключится или “упадёт”, система должна иметь возможность восстановиться или перезапустить его автоматически. Необходимо множество уровней безопасности, чтобы процессы могли наблюдать за исполнением друг друга, следя за ошибками и помогая в восстановлении неисправных элементов.

   Однако и этого недостаточно. Хотя сама нейросеть заботится о своём обучении, включении новых идей, росте и накоплении опыта, существует также нечто, что биологические системы неспособны делать со своим разумом – биологические системы не могут менять свои нейронные структуры. Однако нет никакой необходимость отображать это ограничение в машинах.

5) Напротив, наш ЯП должен позволять “горячую” замену кода [2] для наделения системы возможностью исправлять ошибки и обновлять себя без необходимости уводить что-нибудь в оффлайн.

6) Должен позволять системе выполняться произвольно долго, ошибки должны быть локальны и исправимы самой системой.

Учтём также, что нейросетевые системы должны управлять многими сложными системами, такими как беспилотные летательные аппараты (дроны), и язык должен поддерживать простую разработку многочисленных драйверов для разнообразного железа.

Эрланг был разработан как язык для создания телефонных коммутационных систем. Эти системы предъявляют множество требований; приведём список требований к языку для разработки таких систем [19]:

1) Примитивы инкапсуляции – должно быть некоторое количество механизмов для ограничения последствий ошибки. Должно быть можно изолировать процессы, чтобы они не могли навредить друг другу.

[2] code hot-swapping

2) Параллельность (concurrency) – язык должен поддерживать легковесный механизм создания параллельных процессов и передачи сообщений между процессами. Переключение контекстов между процессами и передача сообщений должны быть эффективными. Параллельные процессы также должны разумно распределять время на ЦПУ, чтобы ни один из процессов его не монополизировал.

3) Примитивы обнаружения неисправностей – позволюящие процессу наблюдать за другим процессом и замечать, что он по какой-либо причине завершился.

4) Прозрачность обнаружения [3] – если мы знаем Pid процесса, мы должны мочь послать ему сообщение.

5) Динамическое обновление кода – должно быть возможно динамически менять код в работающей системе. Отметим, что поскольку многие процессы будут выполнять один и тот же код, нам нужен механизм, позволяющий процессам выполнять “старый” код, а “новым” процессам выполнять изменённый код в то же самое время.

   С набором библиотек для обеспечения наличия и возможности поддержки:

6) Стабильного хранения – такого, которое переживёт падение.

7) Драйверов устройств – они должны предоставлять механизм общения со внешним миром.

8) Обновления кода – позволяющего обновлять код в работающей системе.

9) Инфраструктуры – для запуска/останова системы, ведения лога ошибок и т.д.

Именно эти возможности предоставляет Эрланг, и именно поэтому в этой работе он используется для разработки ИНС.

[3] location transparency

SENNs (ИНС с субстратным кодированием)

В этой главе и в дальнейшем нейроузлы (neural nodes ,neurodes) будут называться “нейроды”.

Рассмотрим устройство субстратно кодированных нейронных сетей более подробно.

Субстратное кодирование позволяет выращенным нейронным системам быть чувствительными к геометрическим образам [4] в сенсорных сигналах. Непрямое кодирование успело проявить свои преимущества в областях обобщений, задач распознавания изображений и задач с геометрическими регулярностями.

Субстратное кодирование позволяет выращивать субстратные модули, нейроды которых управляются прямо кодированной нейронной сетью (ПКНС); проводя эволюцию ИНС, выращивают систему, которая принимает координаты нейродов в субстрате и выдаёт топологию, синаптические веса и параметры связности многомерного субстрата.

Субстратное кодирование было популяризировано подходом HyperNEAT [44]; метод, сравнительно простой, был ответом на известное в области ИИ “проклятье размерности” [5] – количество переменных, с которыми нужно иметь дело при росте размеров системы. В этом методе используется непрямое кодирование, и поэтому входящие сигналы обрабатывает не выращенная ПКНС, а многомерный субстрат, насыщенный нейродами.

Субстрат представляет из себя гиперкуб[6], заданный по каждой оси на отрезке $[-1;1]$. Внутри гиперкуба находятся нейроды, как показано на рис. 5.1. У каждого нейрода есть координаты. Разные нейроды связаны друг с другом.

Однако исследователю не нужно выращивать паттерны связности [7] или синаптические веса, вместо этого их настраивает ПКНС. Таким образом, даже если есть трёхмерный субстрат с миллионами нейродов и множеством связей, в настраивающей ИНС их могут быть всего единицы. Это

[4] geometrical regularity sensitive

[5] curse of dimensionality

[6] В строгом смысле; обычно “гиперкубом” называют четырёхмерный куб. Вообще говоря, геометрия субстрата может и не соответствовать многомерному кубу, но в этой работе это рассматривается не очень подробно.

[7] neurode connectivity patterns

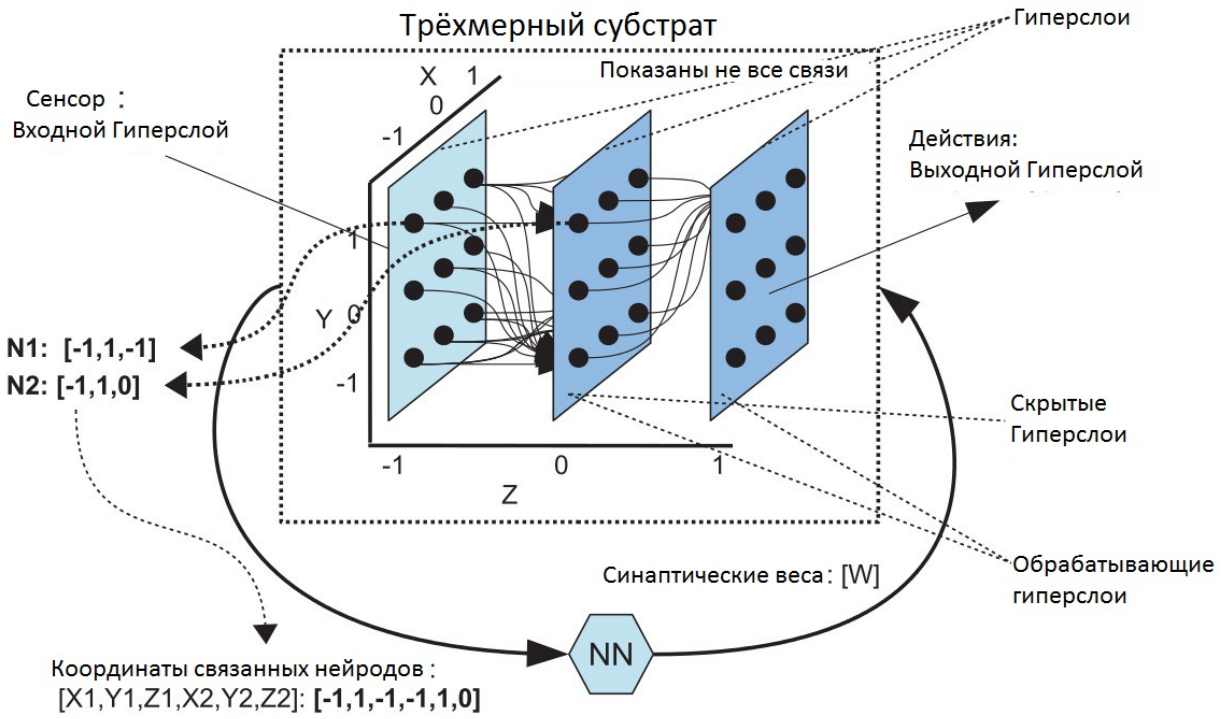

Рисунок 5.1 — Субстратно кодированная сеть. [1]

достигается подачей на вход ИНС координат нейродов и использованием выхода для настройки синаптических весов между ними. Поэтому, сколько бы нейродов ни было в субстрате, эволюция изменяет лишь несколько, содержащихся в прямо кодированной сети (хотя, конечно, чем больше нейронов в такой сети, тем сложнее те связи, которые она может создать в субстрате). Длина входного вектора в таком случае не превосходит 2*(Размерность Субстрата).

В силу того, что ПКНС имеет дело с координатами, она становится чувствительной к геометрическим регулярностям во входных данных. Это позволяет ей настроить топологию субстрата так, чтобы лучше отображать геометрические особенности решаемой задачи.

В HyperNEAT прямо кодированная сеть называется CPPNN (Compositional Pattern Producing Neural Network, нейронная сеть, производящая составные паттерны). Это связано с тем, что в обычном NEAT эволюции подвергаются сети, которые используют в качестве функции активации только tanh (гиперболический тангенс), а в HyperNEAT функции активации разных нейронов могут быть разными.

DXNN2, которая используется в этой работе, поддерживает создание нейронов и нейродов с разными функциями активации. Созданные в рамках DXNN2 ПКНС могут использоваться для любых целей, в том числе и для настройки субстратов. Поэтому в

этой работе отдельный термин для такого использования не вводится.

Нужно определить, какова будет топология субстрата, как передавать ему данные от сенсоров и как использовать его вывод для управления актуаторами. Чаще всего при создании субстратов используется топология стандартного гиперкуба; нейроды каждой гиперплоскости связываются с нейродами следующей, реализую прямое распространение [8]. В другом популяром варианте все нейроды взаимосвязанны [9]. В обоих случаях ПКНС задаёт синаптические веса связанных нейродов. Но мы можем позволить ПКНС выводить не только веса, но и то, есть ли вообще связь между нейродами. Такую топологию называют свободной [10].

На рис. 5.2 показаны примеры субстратов со свободной топологией (справа) и Джордановой рекуррентной (последний гиперслой соединяется с первым).

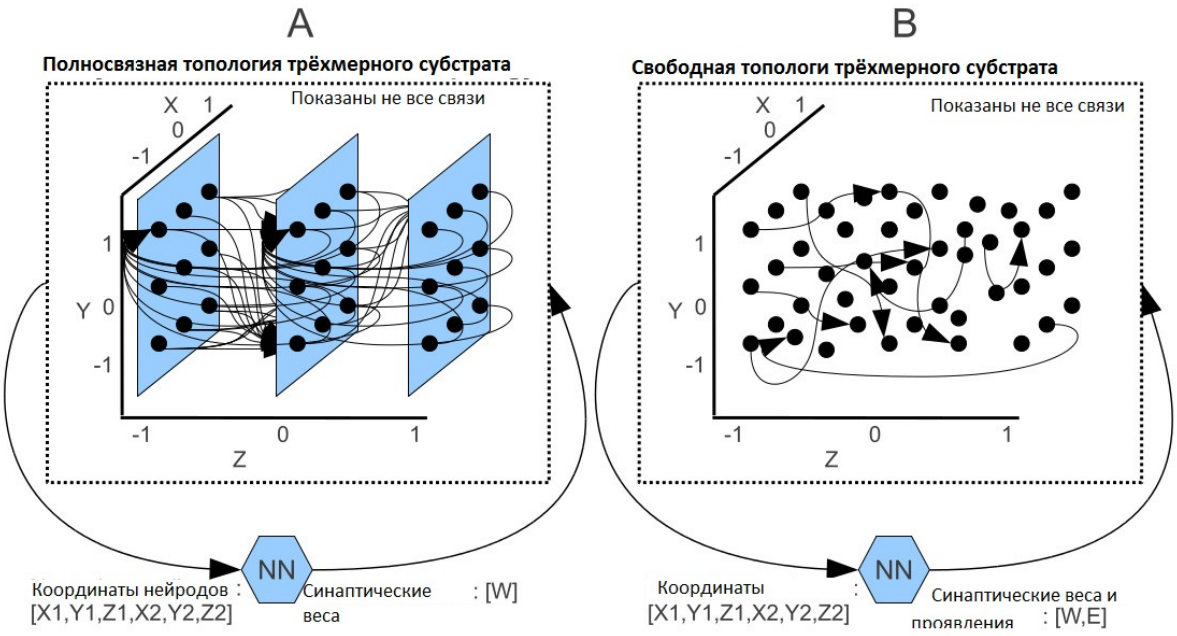

Рисунок 5.2 — Полносвязная и свободная топологии. [1]

Хотя обычно нейроды в субстрате распределены равномерно в пределах гиперслоя, это не обязательно; более того, в некоторых задачах хорошо работает ПКНС, которая со временем создаёт в субстрате зоны различной плотности и соответственно восприимчивости к сигналам.

Гиперслоем называется группа нейродов, имеющих одну, самую внешнюю координату. Так, в пятимерном субстрате (x, y, z, a, b) гиперслои будут четырёхмерными,

[8] feedforward
[9] interconnected
[10] freeform

расположенными на разных координатах b.

Плоскости, составляющие гиперслой, будут называться гиперплоскостями. Их размерность на 1 меньше размерности гиперслоя.

Такая терминология поможет проще описывать используемые субстраты.

Подавать SENN данные также можно по-разному. К примеру, если сенсорным сигналом является изображение и используется трёхмерный субстрат со стандартной топологией прямого распространения, можно расположить изображение в качестве входного гиперслоя на $Z = -1$. Тогда у пикселей будут считаны координаты. Затем ПКНС сможет назначить синаптические веса нейродам в гиперслое $Z = 0$ (см. 5.3).

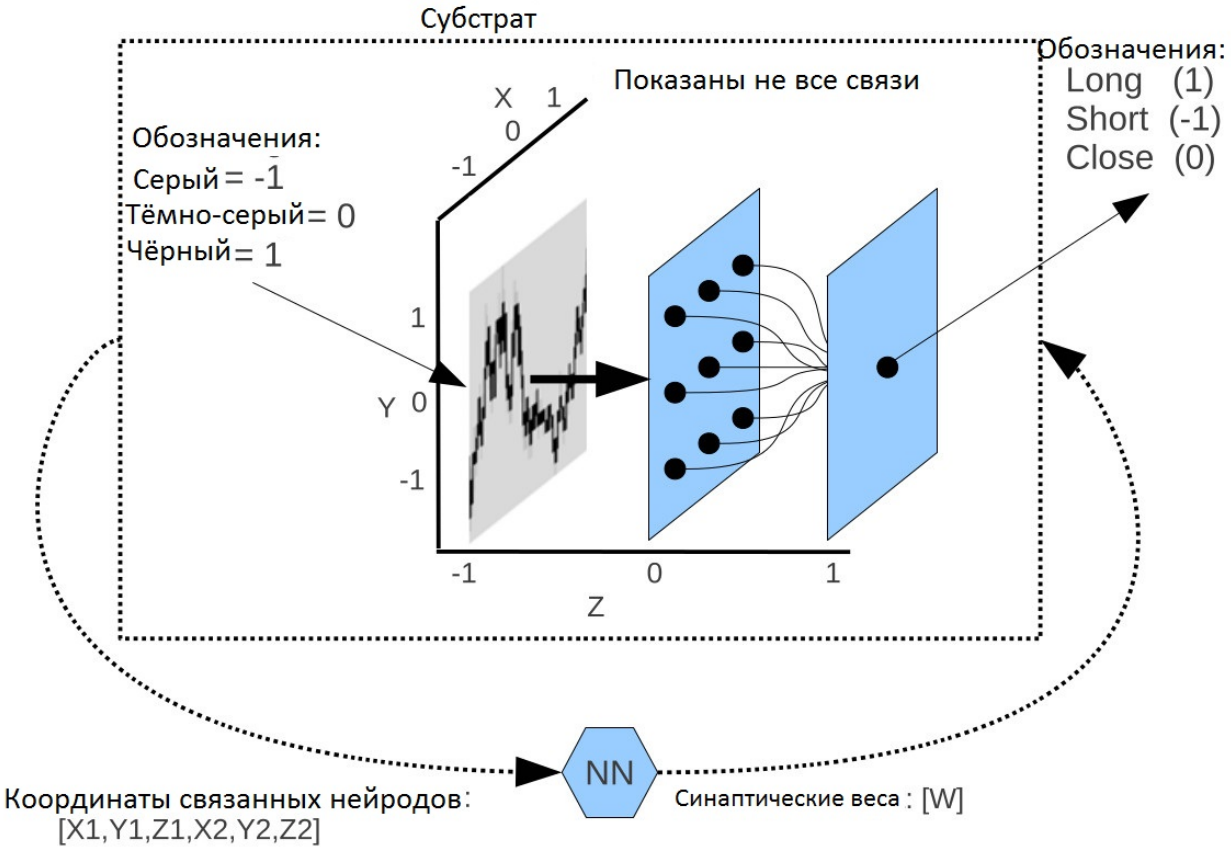

Рисунок 5.3 — Назначение синаптических весов в субстрате с использованием ПКНС. [1]

Так же мы можем назначить нейроды последнего гиперслоя выходными и использовать их выводы как сигналы для управления актуаторами. Какие нейроды для управления каким актуатором назначить может решаться исследователем (см. рис. 5.4).

Сенсоры также могут быть связаны по-разному. Например, четырёхмерный субстрат может иметь трёхмерный входной гиперслой, где каждая

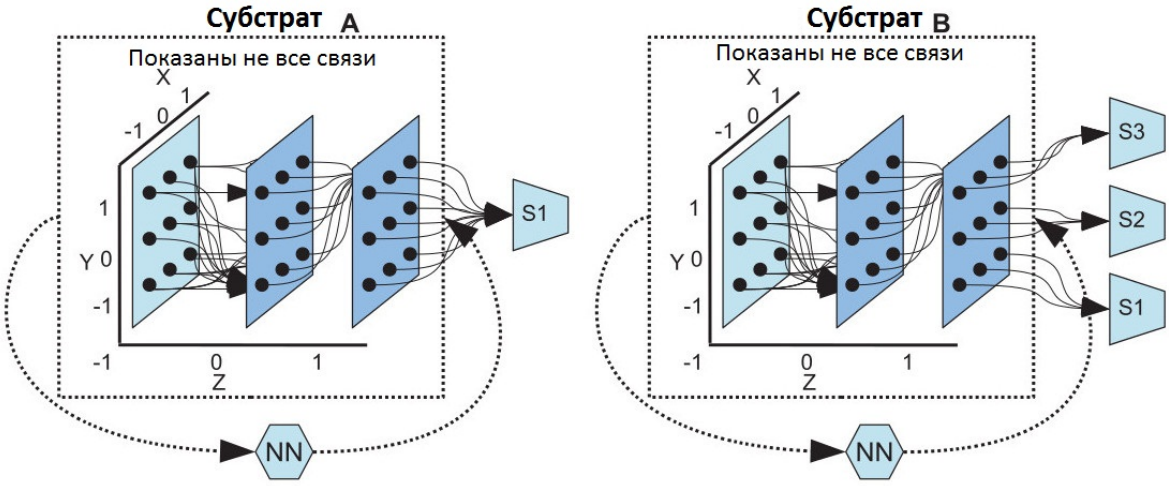

Рисунок 5.4 — На картинке слева все нейроды выходного гиперслоя связаны с одним актуатором, а справа – по три нейрода на каждый из трёх актуаторов. [1]

гиперплоскость – изображение (возможно, с камеры). На картинке (см. рис. 5.5) изображена именно такая сеть, принимающая на вход графики, возможно разного разрешения и использующие разные технические показатели.

Наконец, можно менять геометрию субстрата, чтобы лучше отображать геометрию анализируемых данных. На картинке изображён пример субстрата, для работы с которым ПКНС использует полярные координаты. Сенсорные сигналы подаются на внешнюю окружность, а вывод производится внутренней (см. рис. 5.6).

Как можно видеть, теоретически можно построить любой паттерн связности и назначить любые синаптические веса. Если также учитывать, что искусственный нейрон является универсальным аппроксиматором, можно представить, что можно вырастить субстрат, с достаточной точностью воспроизводящий топологию мозга. Более того, такому субстрату даже не обязательно быть трёхмерным. Но область достоверного воспроизведения человеческих нейронных сетей в этой работе далее не рассматривается.

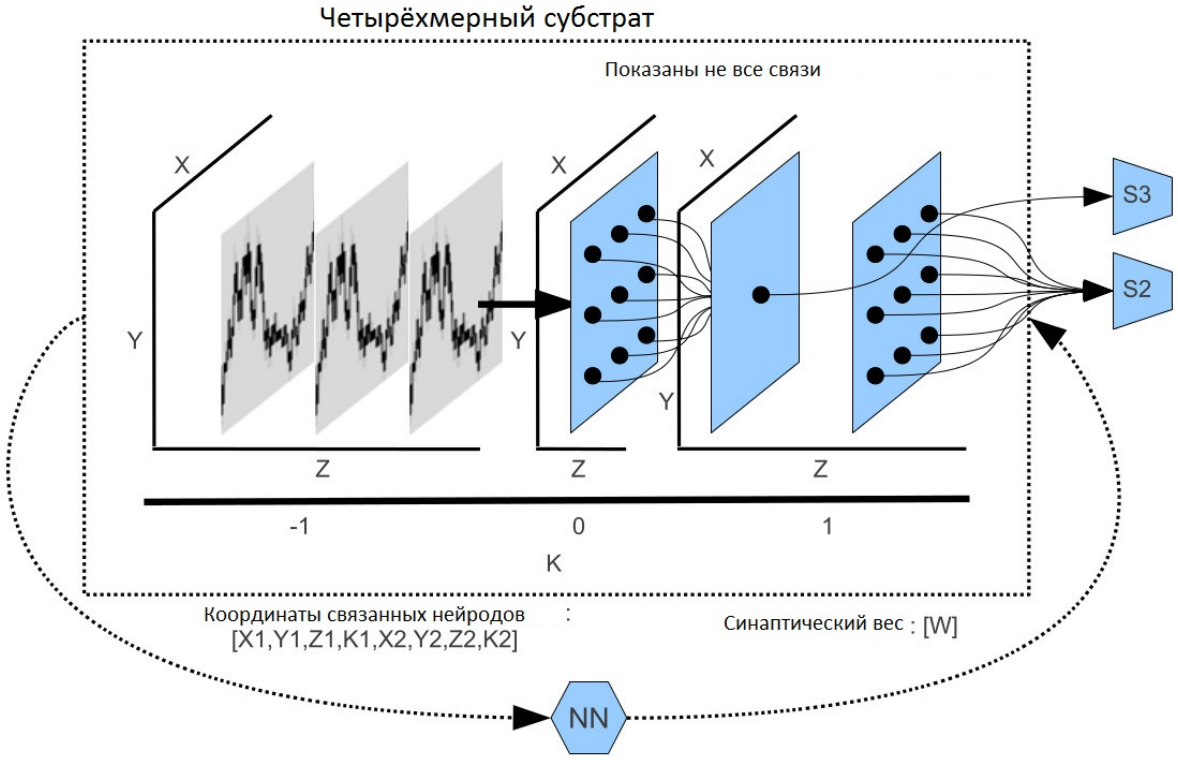

Рисунок 5.5 — Четырёхмерный субстрат. [1]

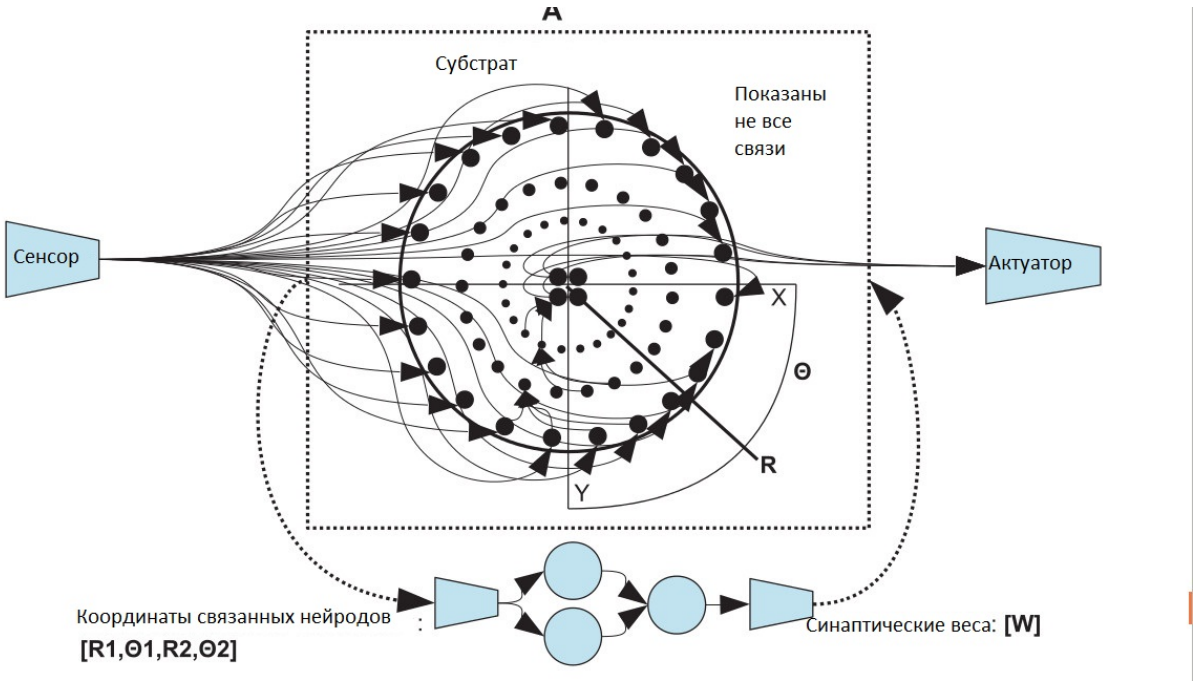

Рисунок 5.6 — Использование полярных координат в субстрате. [1]

Используемый метод с точки зрения нейроэволюции

В английском используется слово chart, которое обычно переводится как “диаграмма, график или схема”. Из картинок будет понятно, что представляют из себя chart’ы, с которыми мы имеем дело. В тексте они будут называться графиками, если из контекста ясно, какой вид отображения данных имеется в виду.

Электронная торговая платформа, в частности весьма популярный MetaTrader или dxTrade [11], предоставляет возможность визуализировать данные в форме “японских свечей” (candlestick chart) [50], на которой отображаются текущая, наибольшая и наименьшая цена на выбранном временном интервале (который также называется шаг [12] или тик [13]). На рис.5.7 представлен пример подобного графика.

Рисунок 5.7 — Интерфейс электронной торговой платформы на примере dxTrade. Из [?]

ИНС с прямым кодированием обычно используют метод плавающего

[11] dxTrade (https://devexperts.com/en/trading-platforms/dxfx/dxtrade.html)
[12] step
[13] tick

окна [14], в котором сети поставляется вектор, список цен. Геометрические особенности в этом векторе не содержатся (см. рис. 5.8).

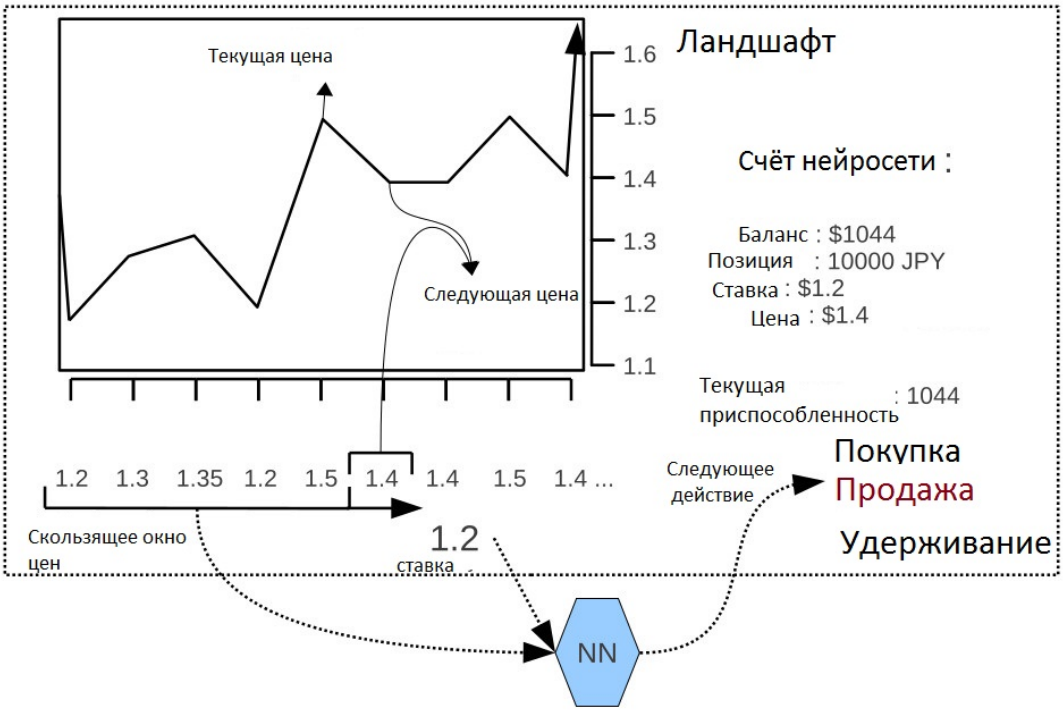

Рисунок 5.8 — Метод скользящего окна. [1]

Как упоминалось выше, можно использовать SENNs для анализа графиков напрямую и использования геометрических образов, которые в них содержатся. Поскольку каждый нейрод субстрата имеет связь с каждым нейродом или элементом ввода предыдущего гиперслоя, график, подаваемый на субстрат, нужно сначала привести к должному разрешению, которое позволит сохранить геометрические образы и при этом будет вычислительно посильно (computationally viable). К примеру, если использовать 1000 исторических моментов, на вход сети будет подано изображения с разрешением 1000x1000 пикселей, и в первом скрытом гиперслое субстрата придётся использовать 1000000 нейродов. Если гиперслои расположить на оси Z следующим образом: при $Z = -1$ вход с миллионом нейродов, при $Z = 0$ на скрытой гиперплоскости 10x10, и на третьем гиперслое при $Z = 0$ 1 нейрод, то на обработку одного “кадра” графика цен понадобится порядка 10^8 вычислений.

На рис. 5.9 изображена другая архитекутра SENN, но принцип тот же.

[14] sliding window approach

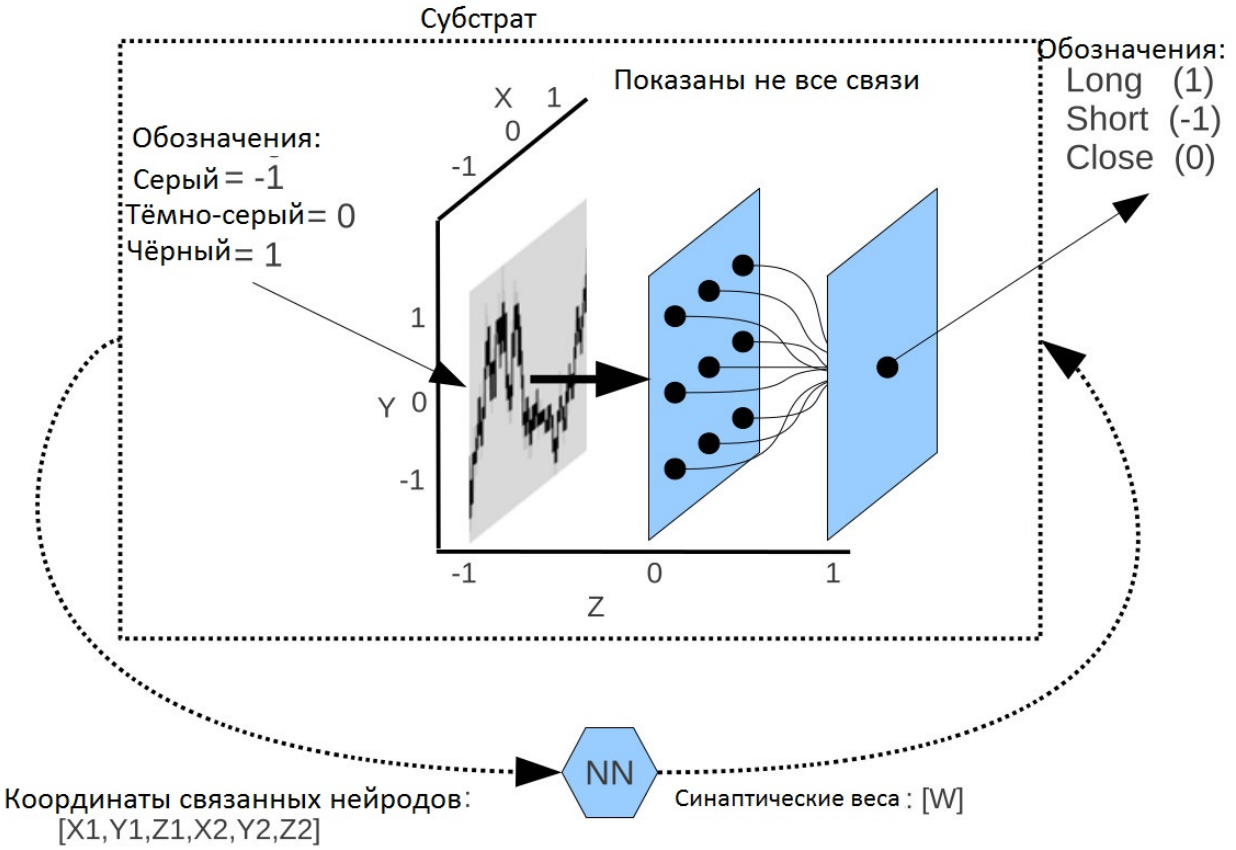

Рисунок 5.9 — Архитектура SENN. [1]

Важно подобрать и проверить разрешение, которое сохраняло бы геометрические образы, но при этом не перегружало бы нейросеть. На рис.5.10 изображены результаты построения графиков с разрешением 100x20 и 10x10. Можно видеть, что геометрические особенности сохраняются.

Когда ИНС (ПКНС и субстрат) построена, нужно создать на её основе нейронную систему; добавить к ней сенсоры, которые будут считывать графики, и актуаторы, которые будут взаимодействовать с электронной торговой платформой и осуществлять торги.

В итоге мы построим два вида нейронных систем – PLI и PCI (Price List Input и Price Chart Input) и испытаем их на приватных ладшафтах симуляции Форекса. Симуляция использует исторические данные о ценах и работает следующим образом: исторические данные (в нашем случае о курсе EUR/USD) будут внесены в ets таблицу. Используется 1000 значений цен с 5 по 20 ноября 2009 и 15-минутные тики, что обеспечивает распределение цен (price spread) в $0.00015, что примерно равно распределению для стандартных учётных записей поставщиков финансовых услуг, таких как Alpari.

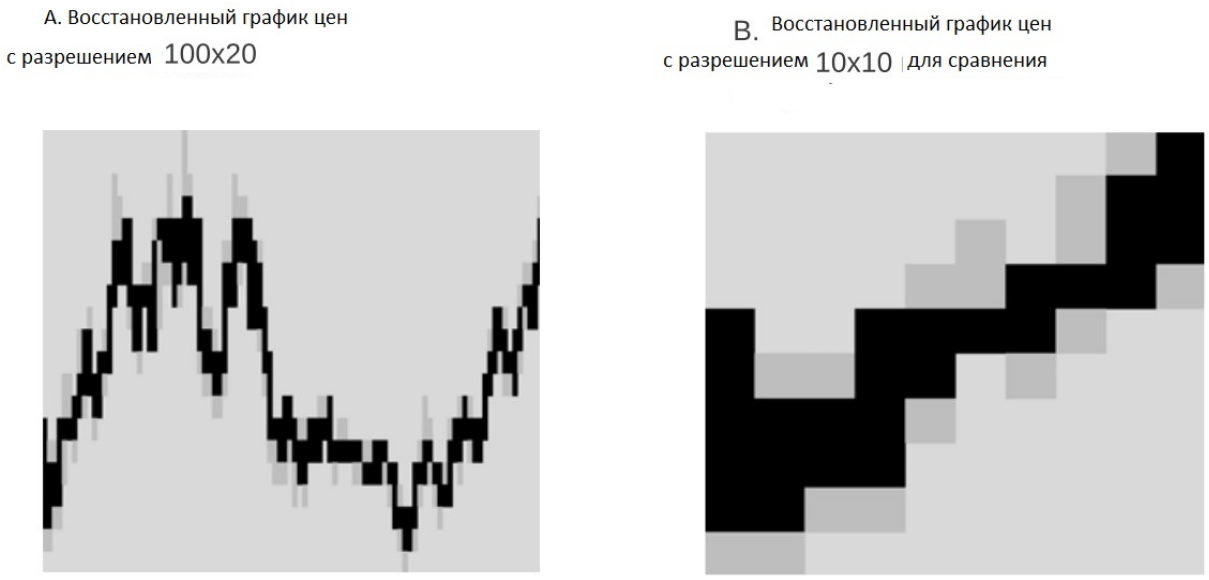

Рисунок 5.10 — Восстановленные графики. [1]

Каждому нейроагенту будет дано $300. Также, каждый раз, когда агент будет открывать позицию, он будет ставить $100, преобразованные в $5000 с помощью финансового рычага [15], что является стандартной практикой. Каждый агент будет взаимодействовать со своим частным ландшафтом, производя для него вывод, который будет преобразован его актуатором в одно из чисел {-1,0,1}, означающие “занять короткую позицию” (go short), “закрыть позицию” (либо ничего не делать, если открытых позиций нет) и “занять длинную позицию”(go long) соответственно.

В ходе эволюции ИНС всегда есть вероятность того, что вместо обучения идее, описывающей данные [16], сеть просто запомнит их. Эта вероятность особенно велика в случае работы со статическим списком данных.

В симуляциях искусственной жизни среда настолько динамична, что запоминание данных (переучивание, overfitting) быстро приведёт к смерти искусственного организма. В обучении же сетей в области финансового анализа обычно используют статический набор данных.

Чтобы проверить способность выращенных агентов к обобщению и работе с ранее не встречавшимися данными, они будут поделены на обучающее и обобщающее подмножества [17]; в первый набор войдут первые 800 цен, в проверочный – остальные 200. Каждое новое поколение или каждые

[15] leverage
[16] concept behind the data
[17] training and generalisation subsets

X оценок в популяции выбирается лучший [18] и применяется к проверочному подмножеству, с которым он ранее не встречался. Если в некоторый момент агент будет показывать себя всё хуже на проверочных данных, но при этом продолжит всё лучше справляться с обучающим множеством, он будет сочтён переобученным и его обучение завершится.

Агенты будут опрашивать свои сенсоры, принимать решения и реализовывать их с помощью своих актуаторов. За сбор статистик будут ответственны отдельные процессы.

Оценка [19] считается случившейся, если агент обработал все 800 точек обучающих данных или его баланс стал ниже $100. Приспособленностью агента считается сумма заработанных им долларов на момент окончания оценки. Каждая эволюционная последовательность [20] будет продолжаться 25000 оценок, и каждый эксперимент состоит из 10 таких последовательностей. В каждом эксперименте размер популяции назначен равным 10. Наконец, в каждом эксперименте ИНС позволяется использовать любую функцию активации из следующих: [**tanh**, gaussian, **sin**, absolute[21], sgn, linear, log, sqrt].

В случае SENNs выход актуатора будет подаваться на входной гиперслой, что делает топологию всего субстрата Джордановой рекуррентной:

Будет проведено 5 экспериментов с ИНС прямого кодирования, использующими списки цен и скользящие окна разного размера. Также будет проведено 9 экспериментов с ИНС на основе 4-мерного субстрата, использующими графики цен в виде восстановленных изображений разного разрешения. Входящий гиперслой будет состоять из сенсоров fx_PCI, fx_internals и Джордановой рекуррентной связи. В четырёхмерном субстрате входной гиперслой будет расположен на $K = -1$ и будет состоять из 3 гиперплоскостей; все они будут связаны с 5x5 гиперслоем на $K = 0$, а он – с 1x1 выходным гиперслоем на $K = 1$, который будет выдавать сигнал о том, какую позицию занимать, а также будет рекурретно отсылать свой сигнал на входной гиперслой.

Способности к обобщению будут проверяться каждые 500 оценок; это

[18] champion
[19] evaluation
[20] evolutionary run
[21] Имеется в виду функция "абсолютное значение"или "модуль"

позволит не только оценить способности лучшей ИНС в популяции к обобщению, но и построить графики этой способности для разных видов ИНС и для всей TWEANNs платформы вообще.

Структура используемой платформы

В используемом методе предполагается, что практически все сущности реализованы с помощью процессов, написанных на Эрланге[1].

Основным "местом"существования всех сущностей является полис [22]. В рамках одного полиса существуют различные агенты – нейронные системы, состоящие из ИНС разной архитектуры с подсоединёнными сенсорами и актуаторами. Агенты, над которыми проводятся эксперименты, взаимодействуют с ландшафтами. За отбор лучших, внесений изменений и запись эволюционной истории ответственны отдельные процессы.

[22] polis

Код

В этой работе используется метод, реализованный Шером [1, 3] на Эрланге. При его первичной проверке на локальной машине использовалась Ubuntu 14.04 x64 и Эрланг версии R16B03. Во время проведения экспериментов использовался более новый на тот момент Эрланг R18B03.

Нейроны реализованы с помощью процессов Эрланга; их общение друг с другом также реализовано стандартными средствами языка. Процессы, следящие за нейронами, полисы, ландшафты и другие процессы наблюдения используют стандартное для Эрланга поведение [23] и другие методы, известные под общим названием OTP [24] . Эволюционные процессы, реализованные в TWEANN плафторме, хранят генотипы особей (в частности, агентов) в базах данных, популярных в среде разработчиков на Эрланге – отчасти ets, но в основном Mnesia [25].

Весь код, реализующий используемые методы, доступен в репозитории: Neuroevolution through Erlang (URL: https://github.com/StrangeTcy/Book_NeuroevolutionThroughErlang/tree/master/Ch_19).

[23] gen_server behaviour // Erlang Documentation.
URL: http://erlang.org/doc/design_principles/gen_server_concepts.html

[24] Erlang/OTP // Official Erlang site. URL: http://www.erlang.org

[25] Mnesia. //Erlang documentation.
URL: http://erlang.org/doc/man/mnesia.html (дата обращения: 13.05.2016)

Выводы

Оценка результатов

Эрланг по природе своей способен поддерживать значительное количество потокообразных сущностей (так называемых акторов [54]), однако фактически скорость выполнения кода, написанного на Эрланге, растёт с ростом количества доступных процессорных потоков (CPU core threads)[26].

Вообще говоря, каждый построенный агент должен быть не очень требователен к ресурсам. Однако же в ходе экспериментов множество агентов подвергалось обработке большое количество раз, а такая задача выигрывает от проведения вычислений на мощных машинах. Эксперименты проводились на 3 предоставленных компанией Google виртуальных машинах [27] с 8 vCPU (один vCPU = гипертред на 2.6GHz Intel Xeon E5 (Sandy Bridge)) и 30 GB RAM. В качестве операционной системы использовалась Ubuntu 14.04 LTS x64; версия Эрланга – R18B03.

Было проведено в общей сложности 14 экспериментов для агентов на основе ИНС двух видов:

1) SlidingWindow5
2) SlidingWindow10
3) SlidingWindow20
4) SlidingWindow50
5) . SlidingWindow100

   и

6) ChartPlane5X10

[26] Этот рост, наверное, более чем линейный. Но отдельных вычислений степени не проводилось

[27] Google Compute Cloud Platform (URL: https://cloud.google.com)

7) ChartPlane5X20
8) ChartPlane10X10
9) ChartPlane10X20
10) ChartPlane20X10
11) ChartPlane20X20
12) ChartPlane50X10
13) ChartPlane50X20
14) ChartPlane100x10.

Для агентов были вычислены значения приспособленности (fitness) в зависимости от количества проведённых оценок (evaluations). Были получены следующие результаты (рис. 5.11, 5.12, 5.13):

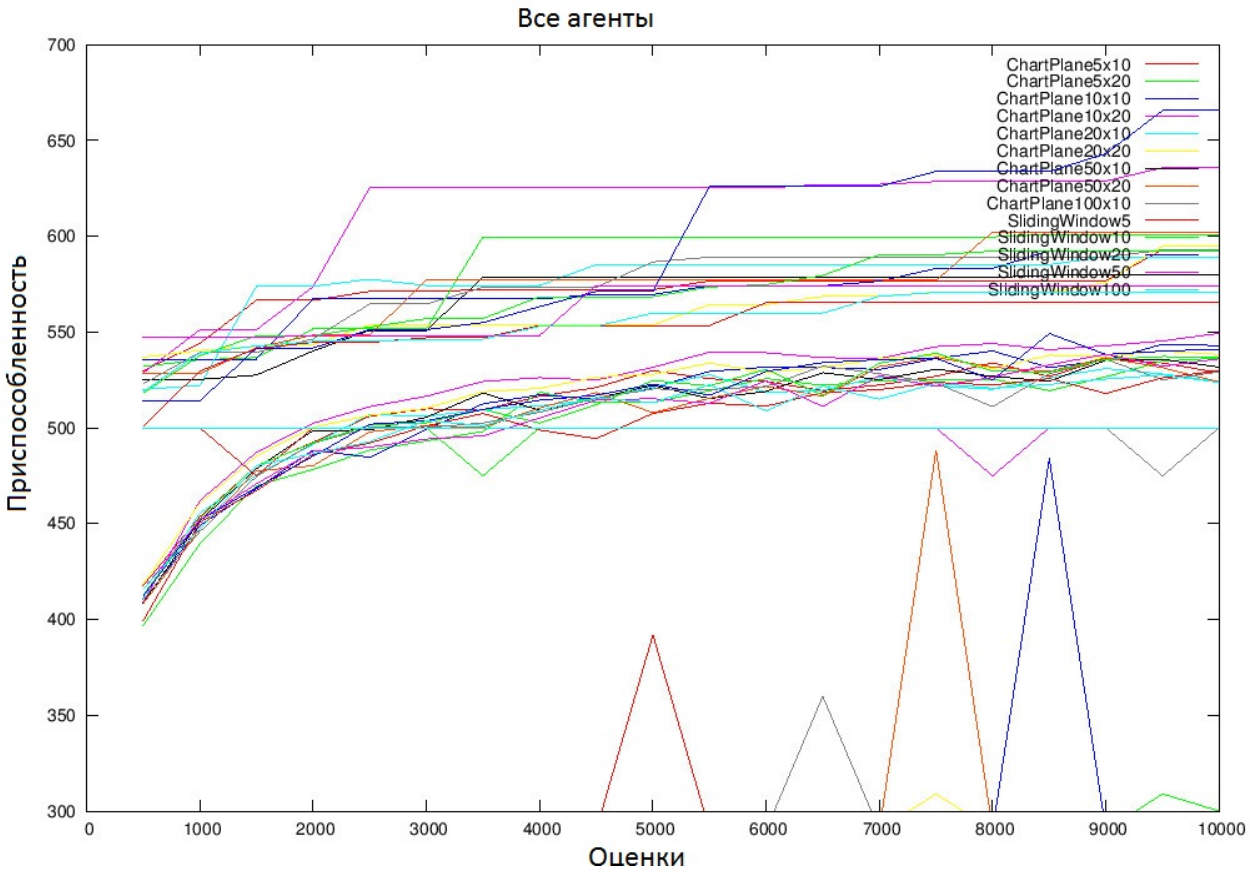

Рисунок 5.11 — Результаты эксперимента для всех агентов.

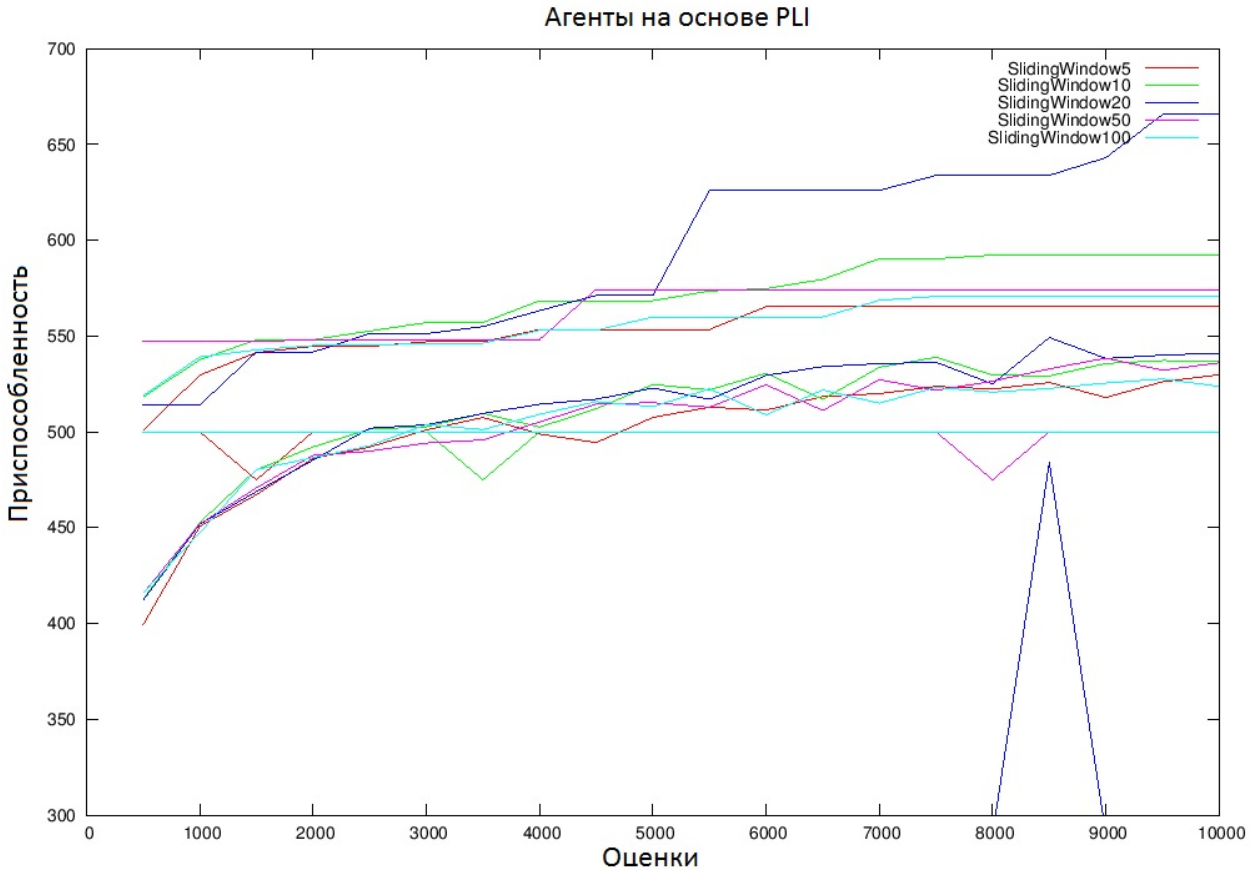

Рисунок 5.12 — Приспособленность для агентов на основе PLI.

Можно видеть, что агенты обоих видов принесли прибыль. Из [1, 3] известно, что для обучения наибольшее возможное значение баланса – \$704, а для проверки обобщения – \$428; относительно этих показателей прибыль, принесённая выращенными агентами, может считаться значительной.

PLI агенты адаптировались довольно быстро и устойчиво, иногда совершая резкие “адаптивные скачки”. В свою очередь, PCI агенты показали более рано начавшийся и более стремительный рост адаптированности, которая со временем стала меняться более плавно. Таким образом, геометрическая чувствительность не только работает, но и показывает осмысленные результаты.

Это позволяет предположить, что PCI агенты быстрее адаптировались, а именно, что они быстрее достигли состояния, в котором небольшая часть популяции удалялась из-за переучивания, и большая часть агентов оставалась в популяции всё дольше. Что позволяет надеяться, что способность PCI агентов к обобщению может быть перенесена [28] на реальные

[28] carried over

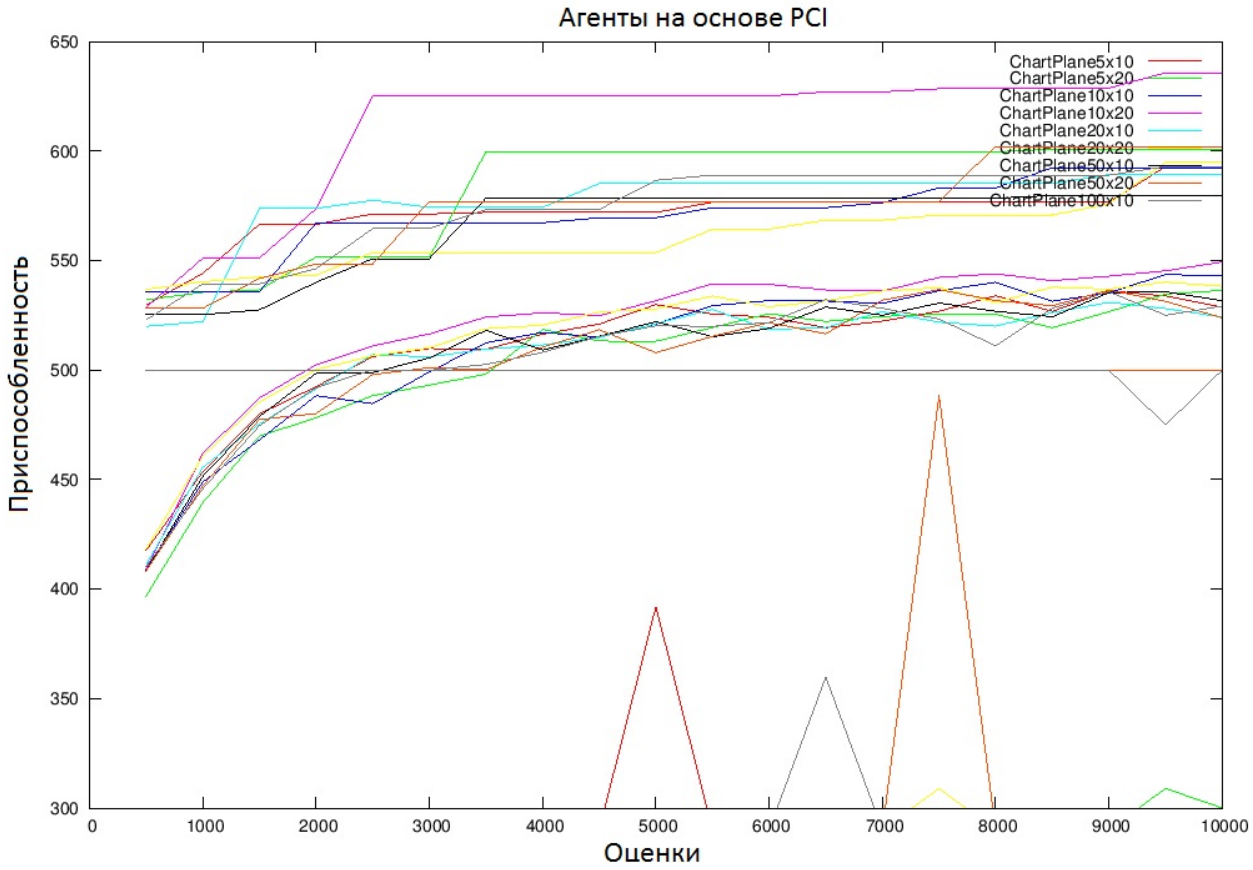

Рисунок 5.13 — Приспособленность для агентов на основе PCI.

финансовые данные.

Дальнейший анализ графиков показывает, что лучше всего себя проявили агенты с разрешениями 5x20 и 10x20 и агенты с размером скользящего окна 10 и 20 соответственно; вопреки ожиданиям того, что увеличение разрешения должно привести к увеличению приспособленности.

Возможным объяснением такого поведения для PCI является следующее: с ростом разрешения входного изображения у нейродов постсинаптического слоя появляется столько входов, что они становятся насыщеными и не могут эффективно работать.

Про PLI можно сказать, что при небольшом размере скользящего окна (10 и 20) в ИНС присутствует некоторое количество рекуррентных связей, мешающих простому переучиванию и запоминанию (memorisation) и побуждающих агентов к обобщению. Однако увеличение количества таких связей, похоже, приводит к уменьшению производительности сети.

Данные гипотезы требуют более тщательной проверки и дальнейших экспериментов, как и вся область нейроэволюции в целом.

Используемая нейроэволюционная платформа поддерживает нейро-

пластичность в ИНС, управляющих агентами, но в данной серии экспериментов она не была использована. Проведение бенчмарков для агентов, использующих нейропластичность, сопоставление результатов с результатами испытаний агентов, которые её не используют, и осмысление сравнения – отдельная исследовательская задача (требующая длительных вычислений и могущая стать целью отдельной (следующей) работы в области нейроэволюции).

В целом можно видеть, что, как и ожидалось, геометрически восприимчивые PCI агенты показали своё превосходство над PLI и свою способность к обобщению и прибыльной торговле. Более того, можно считать, что полученные в результате эксперимента лучшие особи с большой вероятностью (связанной с точностью симуляций Форекса) проявят себя в обработке реальных данных в реальном времени. Также можно отметить, что такие агенты будут пользоваться всеми преимуществами, которые предоставляет Эрланг.

Планы на будущее

> Friendliness is something that human beings are born with.
> Artificial intelligences are born with objectives.
>
> H.Finch[29]

DXNN2, использованная в этой работе TWEANNs платформа, была использована для выращивания и испытания DENNs и SENNs. В проведённых экспериментах последние были использованы для геометрического анализа графиков Форекса, но на самом деле эти ИНС способны к решению произвольных задач анализа временных рядов [30]: предсказанию приливов, землетрясений, солнечных пятен, погоды, человеческой речи и энцефалограмм, и многому другому.

Построенные SENNs и саму платформу можно использовать для построения системы, способной к машинному слуху и машинному зрению [31] для обработки сложных событий [32].

С точки зрения структуры используемой платформы ближайшие решения, которые следует воплотить, – написание драйверов, позволяющих агентам на Эрланге взаимодействовать с электронной торговой платформой. В ходе взаимодействия агенты должны получать данные о ценах в реальном времени, обрабатывать их, восстанавливая графики, использовать свои предсказательные возможности для прогнозирования ближайшего (а со временем, возможно, и не только) будущего, а затем принимать решения и действовать на их основе, осуществляя торги.

---

[29] "Дружественность - то, с чем рождаются человеческие существа.
Искусственные интеллекты рождаются с задачами."
Г. Финч

[30] time series analysis problems

[31] voice analysis & computer vision

[32] complex event processing

Также методами нейроэволюции можно создать сети, которые будут заниматься фундаментальным (так называемым сентиментальным) анализом, в первую очередь новостями, и поставлять результаты нейротрейдерам.

Отдельной задачей является применение методов нейроэволюции для слежения за имеющимися нейротрейдерами и выращивания новых. Для этого предполагается модифицировать используемую платформу, дабы эксперименты, подобные описанным в этой работе, могли проводиться распределённо на нескольких машинах, даже и не особо мощных. Точнее, для этого необходимо реализовать возможность общения полисов и их взаимодействия с общими базами данных.

В целом же, как было сказано выше, нейроэволюция способна решить произвольную формально поставленную задачу и является потенциальным путём к сильному искусственному интеллекту [33][49].

[33]Strong AI / Artificial General Intelligence

Список литературы

1. Gene I. Sher. Handbook of neuroevolution through Erlang. New York: Springer Science + Business Media, 2013.

2. Sher G.I. DXNN Platform: The Shedding of Biological Inefficiencies. Neuron, 1-36 , 2010. Available at: arXiv.

3. Sher G.I. Evolving Chart Pattern Sensitive Neural Network Based ForexTrading Agents. 2012. Available at:arXiv.

4. Versace M., Bhatt R., Hinds O., Shiffer M. Predicting The Exchange Traded Fund DIA With a Combination of Genetic Algorithms and Neural Networks. Expert Systems with Applications 27, pp. 417-425, 2004.

5. Yao J., Poh H.L. Forecasting the KLSE Index Using Neural Networks. IEEE International Conference on Artificial neural networks, 1995.

6. Kimoto T., Asakawa K., Yoda M., Takeoka M. Stock Market Prediction System With Modular Neural Networks. International Joint Conference on Neural Networks 1, 1-6, 1990.

7. Hutchinson J.M., Lo A.W., Poggio T. A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. Journal of Finance 49, pp. 851-889, 1994.

8. Refenes A.N., Bentz Y., Bunn D.W., Burgess A.N., Zapranis A.D. Financial Time Series Modelling With Discounted Least Squares Backpropagation. Science 14, pp. 123- 138, 1997.

9. Li Y., Ma W. Applications of Artificial Neural Networks in Financial Economics: A Survey. 2010 International Symposium on Computational Intelligence and Design, pp. 211-214, 2010.

10. Rong L., Zhi X. Prediction Stock Market With Fuzzy Neural Networks. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, pp. 18-21, 2005.

11. Maridziuk J., Jaruszewicz M. Neuro-Evolutionary Approach to Stock Market Prediction. 2007 International Joint Conference on Neural Networks, pp. 2515-2520.

12. Soni S. Applications of ANNs in Stock Market Prediction: A Survey. Ijcsetcom 2, pp. 71-83, 2005.

13. White H., Diego S. Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns. 1988 IEEE International Conference on Neural Networks, pp. 451-458.

14. Dogac S. Prediction of stock price direction by artificial neural network approach. Master thesis, Bogazici University, 2008.

15. Yamashita T., Hirasawa K., Hu J. Application of Multi-Branch Neural Networks to Stock Market Prediction. pp.: 2544-2548, 2005.

16. Quiyong Z., Xiaoyu Z., Fu D. Prediction Model of Stock Prices Based on Correlative Analysis and Neural Networks. Second International Conference on Information and Computing Science, pp: 189-192 , IEEE, 2009.

17. Risi S., Stanley K.O. Indirectly Encoding Neural Plasticity as a Pattern of Local Rules. Neural Plasticity 6226, 1-11, 2010.

18. Eliezer Yudkowsky. Rationality: from AI to Zombies. MIRI, 2015.

19. Joe Armstrong. Making Reliable Distributed Systems in The Presence of Software Errors. The Royal Institute of Technology Stockholm, Sweden, (PhD thesis), 2003;

20. Gerstner W. Spiking Neurons. In Pulsed Neural Networks, W. Maass & C. M.Bishop, eds. MIT-Press, pp. 3-53, 1998.

21. Ang C.H., Jin C., Leong P.H.W., Schaik A.V. Spiking Neural Network-Based Auto-Associative Memory Using FPGA Interconnect Delays. 2011 International Conference on Field-Programmable Technology, 1-4.

22. Luke S., Hohn C., Farris J., Jackson G., Hendler J. Co-evolving Soccer Softbot Team Coordination with Genetic Programming. Proceedings of

the First International Workshop on RoboCup at the International Joint Conference on Artificial Intelligence 1395: 398-411, 1997.

23. Koza J.R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, , 1992.

24. Koza J.R. Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, 1994.

25. Koza J.R. & al. Genetic Programming. Morgan Kaufmann Publishers. 1998.

26. Koza J.R., Bennett F.H., Andre D., Keane M.A. Genetic Programming III: Darwinian Invention and Problem Solving. Springer, 1999.

27. Koza J.R., Keane M.A., Streeter M.J., Mydlowec W., Yu J., Lanza G. Genetic Programming: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, Springer, 2003.

28. Koza J.R., Keane M.A., Yu J., Bennett F.H., Mydlowec W. Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming. Genetic Programming and Evolvable Machines 1, 121-164, 2000.

29. Back T., Hoffmeister F., Schwefel H.P. A Survey of Evolution Strategies. In Proceedings of the Fourth International Conference on Genetic Algorithms, / eds. L. B. Belew and R. K. Booker. Morgan Kaufmann, 1991. pp. 2-9.

30. Holland J.H. Adaptation in Natural and Artificial Systems. J. H. Holland, ed. University of Michigan Press, 1975.

31. Cramer N.L. A Representation for the Adaptive Generation of Simple Sequential Programs. In Proceedings of an International Conference on Genetic Algorithms and the Applications, pp. 183-187. J. J. Grefenstette, ed. Lawrence Erlbaum Associates, 1985.

32. Fogel L.J., Owens A.J., Walsh M.J. Artificial Intelligence through Simulated Evolution. L. J. Fogel, A. J. Owens, & M. J. Walsh, eds. John Wiley & Sons. 1966.

33. Kassahun Y., Sommer G. Efficient Reinforcement Learning Through Evolutionary Acquisition of Neural Topologies. In Proceedings of the 13th European Symposium on Artificial Neural Networks ESANN 2005. ACM Press, pp. 259-266.

34. Siebel N.T., Sommer G. Evolutionary Reinforcement Learning of Artificial NeuralNetworks. International Journal of Hybrid Intelligent Systems 4, 171-183, 2007.

35. Stanley K.O., Risto M. Efficient Reinforcement Learning through Evolving Neural Network Topologies. In Proceedings of the Genetic and Evolutionary Computation Conference, 2002.

36. Robyn M. Dawes, Rational Choice in An Uncertain World, pp. 55-56. 1st ed., ed. Jerome Kagan. San Diego, CA: Harcourt Brace Jovanovich, 1988.

37. Dawkins R. The Selfish Gene. Oxford University Press, 1976.

38. Qingxiang W., T M.M., Liam P.M., Rongtai C., Meigui C. Simulation of Visual Attention Using Hierarchical Spiking Neural Networks. ICIC. pp. 26-31, 2011.

39. Rojas R. 1996 Neural Networks: A Systematic Introduction. Springer, 1996.

40. Gupta M.M., Jin L., Homma N. Static and Dynamic Neural Networks From Fundamentals to Advanced Theory. John Wiley & Sons, 2003.

41. Back T., Hoffmeister F., Schwefel H.P. (1991) A Survey of Evolution Strategies. In Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 2-9. L. B. Belew & R. K. Booker, eds. Morgan Kaufmann, 1991.

42. Moscato P. (1989) On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards memetic Algorithms. 1989. (citeseerx).

43. Krasnogor, N. Coevolution of Genes and Memes in Memetic Algorithms. Proceedings of the 1999 Genetic And Evolutionary Computation Conference Workshop Program, 1999-1999.

44. Gauci J., Stanley K. Generating Large-Scale Neural Networks Through Discovering Geometric Regularities. Proceedings of the 9th annual conference on Genetic and evolutionary computation. GECCO 07, 997. 2007.

45. Risi S., Stanley K.O. Indirectly Encoding Neural Plasticity as a Pattern of Local Rules. Neural Plasticity 6226, 1-11, 2010.

46. Haasdijk E., Rusu A.A., Eiben A.E. HyperNEAT for Locomotion Control in ModularRobots. Control 6274, 169-180, 2010.

47. Coleman O.J. Evolving Neural Networks for Visual Processing. Undergraduate Honours Thesis (Bachelor of Computer Science), University of New South Wales, 2010.

48. Hornik K., Stinchcombe M., White H. Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2, 359-366, 1989.

49. Nick Bostrom. SuperIntelligence: Paths, Dangers, Strategies. Oxford University Press, 2014.

50. Японские свечи. // Википедия, свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Японские_свечи. (дата обращения: 11.05.2016).

51. Head & Shoulders // Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Head_and_shoulders_(chart_pattern). (дата обращения: 11.05.2016).

52. Neuron // Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Neuron. (дата обращения: 11.05.2016).

53. Chart pattern // Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Chart_pattern. (дата обращения: 11.05.2016).

54. Actor Model // Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Actor_model. (дата обращения: 11.05.2016).

Приложение

В английской литературе вся область называется machine learning – “машинное обучение”. Процесс обучения ИНС называется также learning, а само насыщение нейронных сетей знаниями – training (“воспитание”).

Под агентами (agents) понимается деятель как абстрактная сущность, способная совершать действия (actions). В этом смысле и люди, и машины, совершающие торговые действия, являются торговыми агентами (trading agents).

Области, к которым можно отнести нейроэволюцию, называются поразному, и переводы этих названий также разнятся; в тексте могут использоваться переводы английских artificial intelligence, machine intelligence и computational intelligence – “искусственный интеллект”, “машинный интеллект” и “вычислительный интеллект” соответственно.

Общая цель исследований в области AI/ИИ (искусственного интеллекта) – создание artificial general intelligence, общего / универсального / сильного искусственного интеллекта. Эта цель очень интересна, но выходит за рамки этой работы[34].

Как это принято, искусственная нейронная сеть (artificial neural network) будет иногда сокращаться до ИНС, нейронной сети и даже нейросети. Иногда будет использоваться английское сокращение, ANN (ANNs во множественном числе, соответственно).

Системы, основанные на ИНС и обладающие какой-то мерой ИИ, называются neurocognitive systems, что будет переводится как нейрокогнитивные системы.

В ходе нейроэволцюии нейронные сети рассматриваются как особи, которые дают потомство и эволюционируют. Для решения поставленной задачи используется эволюция – we evolve a neural network. Слово “эволюционировать” обычно означает “изменяться в ходе эволюции, не управляемой извне”. Для перевода английского термина будет пользоваться слово “выращивать”. Насколько мне известно, оно не является общепризнанным

[34]“Friendly AI is an extremely tough problem, so people solve it extremely fast.” [18] со ссылкой на [36]

термином в среде машинного обучения и искусственного интеллекта, но оно будет использоваться ради краткости и удобства.

Термином “обучение с учителем” обычно переводят английское “supervised learning” – “обучение с присмотром / наблюдением”.

“Производительность” в данном случае является стандартным переводом английского “performance” и имеется в виду не как мера качества работы железа, а как мера качества работы нейронной сети, того, насколько “хорошо” она решает задачу.

Под машинами в этой работе подразумевается те, которые подвергаются обучению (machine learning) – универсальные электронные вычислительные машины. Никакая конкретная архитектура или топология (отдельный ПК, распределённая сеть, кластер, и т.д.) термином не подразумевается.